Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Ільїна Олексія Володимировича*
(ПІБ)

академічної групи *122-18ск-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка веб-орієнтованної інформаційної системи обліку надання комунальних послуг з використанням Salesforce*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Гуліна І.Г.*			
розділів:				
спеціальний	*доц. Гуліна І.Г.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18ск-2* *Ільїна Олексія Володимировича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-орієнтованної інформаційної системи обліку надання комунальних послуг з використанням Salesforce*

затверджена наказом ректора НТУ «ДП» від «07» червня 2021 р. № 317-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ *доц. Гуліна І.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Ільїн О.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 107 с., 13 рис., 3 дод., 30 джерел.

Об`єкт розробки: веб-орієнтована інформаційна система обліку надання комунальних послуг з використанням Salesforce.

Мета кваліфікаційної роботи: створення єдиного зручного веб-сервісу для сплати різних типів комунальних послуг, відстеження заборгованностей з кожної окремої послуги, внесення актуальних показників лічильників та перевірка актуальних тарифів на комунальні послуги.

У вступі аналізується сучасний стан проблеми, уточнюється мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та конкретизується постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні додатка, що надає можливість електронного зберігання даних про товари, продажі, надходження, ведення бази даних, підбиття підсумків з продажу, отримання звітів і формування супутньої ділової документації.

Актуальність інформаційної системи визначається великим попитом на користування безконтактними методами сплати у час пандемії, окрім підвищення безпеки користування, ця розробка допомогає у відстеженні актуальних показників лічильників.

Список ключових слів: SALESFORCE, КОМУНАЛЬНІ ПОСЛУГИ, ІНФОРМАЦІЙНА СИСТЕМА, ОБЛІК, СПЛАТА, ЛІЧИЛЬНИК, ТАРИФ, ДОДАТОК.

ABSTRACT

Explanatory note: 107 p., 13 figs., 3 app., 30 sources.

Object of development: web-oriented information system of accounting for the provision of utilities using Salesforce.

The purpose of the qualification work: creation of single user-friendly web service for the payment of various types of services, tracking debts for each service, entering the current meter readings and checking the current service's tariffs.

The introduction analyzes the current state of the problem, clarifies the purpose of the qualification work and the scope of its application, provides a justification for the relevance of the topic and specifies the task.

In the first section the subject branch is analyzed, the urgency of the task and the purpose of development are defined, the statement of the task is formulated, the requirements to software realization, technologies and software are specified.

The second section analyzes the existing solutions, selected platforms for development, designed and developed the program, describes the program, algorithm and structure of its operation, as well as calling and loading the program, determines the input and output data, describes the parameters of technical means.

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance lies in the creation of an application that provides the ability to electronically store data on goods, sales, receipts, database maintenance, summarizing sales, receiving reports and the formation of related business documents.

The relevance of the information system is determined by the high demand for the use of contactless payment methods during the pandemic, in addition to improving the security of use, this development helps to track the actual performance of meters.

Keywords: SALESFORCE, UTILITIES, INFORMATION SYSTEM, ACCOUNTING, PAYMENT, COUNTER, TARIFF, APPLIACTION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних;

SOQL - Salesforce Object Query Language;

SOSL - Salesforce Object Search Language;

API - Application Programming Interface;

VF - Visualforce;

CSS - Cascading Style Sheets;

DML - Data Manipulation Language;

ETP - Easy to pay (назва розробки);

REST - Representational State Transfer;

MVC - Model – View – Controller;

HTML - Hypertext Markup Language.

ВСТУП

Розроблена інформаційна система призначена для застосування у сфері державних послуг.

Комунальні послуги – це результат господарської діяльності, що має ціль задовольнити потреби фізичної або юридичної особи у забезпеченні: газом та електроенергією, каналізацією, гарячою та холодною водою, опаленням та утилізацією побутових відходів відповідно діючому законодавству.

Вид розрахунку та вартість послуг визначається постачальником тієї чи іншої послуги, проте, базуючи вартість на офіційних державних тарифах за кожен окремий тип послуги, так як опалення, газ, вода та її відведення.

За умов інформаційної розвиненості нашої планети буває важко зрозуміти необхідність використовувати паперові бази даних, архіви, квітанції та інші застарілі фізичні носії даних. Так, при розрахуванні з комунальними сервісами часто доводиться мати справу, у найкращому віпадку з купою різних веб-систем призначених для сплати, або, у гіршому та частішому сценарії – звертатися за випискою сплати, щодо комунальних послуг, що не є зручним способ відстеження прогресу сплати за комунальні сервіси. Тому, створення єдиного веб-сервісу для сплати, відстеження заборгованностей, внесення актуальних показників та перевірка актуальних тарифів комунальних послуг є об’єктивно необхідним.

Крім простого логічного спрощення сплати кожного з комунальних кослуг, необхідно звернути увагу на загрозу пандемії коронавірусу. Деякі підприємства, що постачають комунальні послуги, для зняття показників надсилають співробітників, які звертаються до кожної житла, з метою фіксування актуальних показників лічильників. Відповідно, для виключення необхідності наражати громадян на ризик бути зараженим, необхіден сервіс, який дозволить вносити показники власноруч.

Так само, наявність у сервісі можливості виконувати платежі, відповідно до заборговоностей за лічильниками, також виключає зараження вірусом, через банківські термінали, якими щодня користується безліч людей.

Таким чином, проблема розглянута в даній кваліфікаційній роботі, є актуальною та має широке практичне значення.

Метою кваліфікаційної роботи є розробка веб-орієнтованого сервісу для обліку й сплати комунальних послуг.

Для досягнення поставленої мети необхідно вирішити основні завдання:

- аналіз найпоширеніших комунальних послуг;
- визначення системи нарахування та розрахунку для кожного виду послуг;
- розробка бази даних, структури програми та її реалізція;

У відповідності до проведенного аналізу визначені основні задачі перед системою:

- спрощення ведення обліку комунальних послуг;
- гнучке керування тарифами на кожну з послуг;
- зручне внесення нових показників лічильника;
- можливість сплатити загальну заборгованність;
- можливість реєстрації на сервісі;
- отримання електроної квітанції про сплату на електрону пошту.

Система забезпечує спрощений погляд на комунальні послуги, коли всі сервіси можно буде сплатити з одного місця.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Комунальні послуги – це життєво необхідний сервіс, що має сплачувати кожен повнолітній громадянин держави, тому у епоху діджеталізації необхідно забезпечити платників зручним та універсальним сервісом для виконання необхідних операцій с постачальниками таких послуг.

Серед постачальників комунальних послуг можна відзначити багато компаній:

- постачальниками води по всій Україні є 81 компанія;
- газ по країні постачають 40 компаній;
- електроенергію постачають 345 компаній.

Зважаючи на кількість існуючих підприємств, набагато зручнішим було б існування одного сервісу, де можна сплатити будь-яку з цих послуг будь-якої компанії.

На сьогоднішній день існує багато непрямих аналогів, що дозволяють сплачувати комунальні послуги. Під непрямими аналогами розуміється банківська система. Майже кожен банк має власний додаток та веб-сервіс, де можна сплачувати комунальні послуги за допомогою номеру договору. Проте вони не мають важливості відслідковувати показники лічильників та слідкувати за актуальними показниками постачальника, окрім того, державний банк «ПриватБанк» навіть утримує деяку комісію при виконанні відповідних платежів.

Найближчим аналогом є компанія Yasno. Yasno є новим брендом компанії «ДТЕК». ДТЕК - найбільша приватна вертикально-інтегрована енергетична компанія України, зо була заснована у 2002 році, що надає послуги у сфері електроенергетики та з недавнього часу газопостачання.

Yasno має власний веб-сервіс та мобільний додаток. Так як компанія займається постачанням декількох видів послуг (електроенергія та газ) через їхній сервіс можна вносити показники лічильників, бачити розмір заборгованості та актуальні тарифи, сплачувати заборгованість у самому сервісі.

Кожен окремий постачальник відповідає лише за оплату власних послуг. Таким чином, універсальність сервісу повністю відсутня і для користувачів це дуже незручно.

Сервіс від підприємства Yasno є одним з найкращих, діючих на даний момент. Він охоплює майже увесь спектр поставлених завдань, але для повноти не вистачає можливості сплачувати за споживання води та зв'язку з іншими постачальниками послуг.

Але ж жодна компанія-постальник комунальних послуг не зможе об'єднати в собі можливості сплачувати всі необхідні послуги. Таким чином, лише підприємство-посередник може надати зручний для користувачів сервіс, який об'єднає в собі можливість сплачувати будь-які послуги.

1.2. Призначення розробки та галузь застосування

Як об'єкт розробки веб-орієнтованої інформаційної системи обліку надання комунальних послуг з використанням Salesforce розглядається веб-сервіс «Easy To Pay», в якому користувач може знайти актуальні тарифи для кожного з видів комунальних послуг (вода, світло, природний газ), окрім того, користувач має можливість вносити показники лічильників, сплачувати заборгованність, отримувати квітанцію після сплати, відслідковувати історію власних платежів. Додатково, користувач може звернутись до свого поточного менеджера, що слідкує за станом платежів цього користувача.

Для адміністрування використовується стандартний інтерфейс організпції Salesforce, що дозволяє мати прямий доступ до бази даних, керувати існуючими користувачами та створювати або призначати менеджера району.

Frontend частина веб-сервісу призначена для:

- відображення корисної інформації у залежності від типу користувача (менеджер або користувач);
- зручного опрацівання запитів користувачів з боку менеджера;
- відображення та керування адміністративними функціями сервісу.

1.3. Підстава для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу. Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки «виконання кваліфікаційної роботи» є:

- освітня програма 122 «Комп'ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» №317-с від 07.06.2021р.;
- завдання на кваліфікаційну роботу на тему «Розробка веб-орієнтованої інформаційної системи обліку надання комунальних послуг з використанням Salesforce».

1.4. Постановка завдання

Завданням кваліфікаційної роботи є розробка веб-орієнтованої інформаційної системи обліку надання комунальних послуг з використанням Salesforce.

Програмне забезпечення призначене для надання універсального інструменту для працювання веб-сервісу.

Програма повинна реалізувати наступні функції:

- легке та доступне адміністрування веб-сервісу;

– формування квітанцій про сплату за комунальні послуги;
– доступ до актуальних тарифів постачальників комунальних послуг;
– надати можливість менеджеру віслідковувати кожного користувача, що з ни пов'язаний;
– контакування з менеджером регіону;
– сплата заборгованностей, згідно показникам лічильників.

Для досягнення поставленої мети необхідно:

– вивчити предметну галузь розв'язуваної задачі;
– створити алгоритм для реалізації поставленого завдання;
– створити базу даних.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для досягнення поставлених цілей програмне забезпечення, що розробляється, повинно підтримувати виконання наступних дій:

– надання віддаленого доступу до веб-сервісу через веб-браузер на комп'ютері користувача;
– зчитування вхідних даних з пристроїв, хмарних сервісів, за адресними посиланнями;
– зберігання даних підсистеми в реляційній базі даних.
– для виконання перерахованих вище функцій у застосунку повинні бути реалізовані:
– можливість інтеграції платіжними системами;
– можливість отримати доступ до веб-сервісу через веб-браузер;
– наявність типової конфігурації, що забезпечує можливість швидкого введення веб-сервісу в експлуатацію;
– програмно-апаратна переносимість.

1.5.2 Вимоги до інформаційної безпеки

Для уникнення некоректної роботи програми необхідно реалізувати:

- семантичний та синтаксичний контроль вхідних даних;
- обробку виняткових ситуацій;
- виведення повідомлень про помилки;
- можливість повторного введення даних;
- платформну незалежність.

1.5.3 Вимоги до складу та параметрів технічних засобів

Для нормального функціонування данного веб-сервісу не важлива технічна складова комп'ютеру, що буде використовуватись для доступу до сервісу, необхідне лише стабільне підключення до мережі Інтернет.

1.5.4 Вимоги до інформаційної та програмної сумісності

Для нормального функціонування програми необхідно, щоб програмне забезпечення персонального комп'ютера, на якому буде функціонувати веб-орієнтована система, відповідало наступним вимогам:

- операційна система Windows (7+), Linux, MacOs;
- веб-браузер Firefox / Google Chrome / Opera / Microsoft Edge / Safari.

Frontend частина додатку має бути реалізована на мові програмування Apex з використанням Visualforce. Робота має бути створена за допомогою компонентного підходу та solid архітектури.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

За завданням кваліфікаційної роботи було реалізовано розробку веб-орієнтованої інформаційної системи обліку надання комунальних послуг з використанням Salesforce..

Призначення розробленої системи:

- оптимізувати облік надання комунальних послуг;
- забезпечити просте адміністрування веб-сервісу;
- надати можливість сплатити заборгованність згідно показникам лічильників.

Розроблена програма реалізує наступні функції:

- легке та доступне адміністрування веб-сервісу;
- формування квітанцій про сплату за комунальні послуги;
- доступ до актуальних тарифів постачальників комунальних послуг;
- надати можливість менеджеру віслідковувати кожного користувача, що з ни пов'язаний;
- контакування з менеджером регіону;
- сплата заборгованностей, згідно показникам лічильників.

Для досягнення поставленої задачі розроблене програмне забезпечення підтримує виконання таких операцій:

- надання віддаленого доступу до застосунку через веб-браузер на комп'ютері або іншому девайсі користувача;
- зчитування вхідних даних з хмарних сервісів та бази даних;
- формування квітанції про сплату та надсилання на пошту користувача;
- коригування складу менеджерів через адміністративну частину.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної галузі розв'язуваної задачі не передбачають застосування математичних методів, при розробці системи відображення та управління контенту інтернет-магазину математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

Веб-сервіс було розроблено за допомогою сучасної технології – Salesforce.

Одноіменна компанія Saesforce - це американська компанія, що базується на хмарному програмному забезпеченні, головний офіс якої знаходиться в Сан-Франциско, штат Каліфорнія. Він забезпечує послугу управління взаємовідносинами з клієнтами (CRM - customer relationship management), а також надає додатковий набір корпоративних програм, орієнтованих на обслуговування клієнтів, автоматизацію маркетингу, аналітику та розробку додатків.

Компанія була заснована 3 лютого 1999 року колишнім керівником Oracle Ромео Ретуерто-молодшим разом із Саудом Убпоном, Барул Олом Унгадом та Франком Домінгуесом як компанія, що займається програмним забезпеченням як послуга (SaaS), і була відкрита публічно в період з вересня по листопад 1999 року.

У червні 2004 року компанія здійснила первинне публічне розміщення акцій на Нью-Йоркській фондовій біржі під акцією CRM та зібрала 110 млн. Доларів США Серед перших інвесторів - Ларрі Еллісон, Магдалена Єсіль, Хелсі Мінор, Стюарт Хендерсон, Марк Іскаро та Ігор Сілл, член-засновник Женевських венчурних партнерів.

У жовтні 2014 року Salesforce оголосив про розробку своєї Платформи успіху клієнтів, щоб поєднати послуги Salesforce, включаючи продажі, сервіс, маркетинг, аналітику, спільноту та мобільні додатки. У жовтні 2017 року

Salesforce запустив інструмент Facebook Analytics для маркетологів від бізнесу до бізнесу. У вересні 2018 року Salesforce співпрацює з Apple з метою вдосконалення програм для бізнесу.

У лютому 2020 року співголова виконавчого директора Кіт Блок покинув посаду в компанії. Марк Беніофф залишився головою та виконавчим директором.

1 грудня 2020 року було оголошено, що Salesforce придбає Slack за 27,7 млрд доларів.[30]

Переваги Salesforce:

Salesforce надає вам найшвидший шлях від ідеї до готового рішення. Ви можете зосередитись на створенні свого додатка за допомогою інструментів Salesforce, а не на побудові інфраструктури та інструментів самостійно. Це може заощадити роки та мільйони доларів.

Клієнти Salesforce зазвичай кажуть, що це унікально з трьох основних причин:

Швидке розгортання традиційного програмного забезпечення CRM може зайняти більше року, порівняйте це з місяцями чи навіть тижнями в Salesforce.

Легко у Salesforce можна витратити більше часу на використання і менше часу на його з'ясування.

Ефективний - оскільки він простий у використанні та може бути налаштований відповідно до бізнес-потреб, клієнти вважають Salesforce дуже ефективним.

Salesforce знаходиться в хмарі, тому ваша команда може використовувати його з будь-якого місця, маючи доступ до Інтернету.

Salesforce повністю масштабована для зростання бізнесу.

Salesforce легко інтегрується зі сторонніми програмами. Якщо ви хочете інтегрувати Salesforce з Gmail, ви можете це зробити, якщо ви хочете інтегрувати його зі своїм бухгалтерським програмним забезпеченням, ви можете зробити це теж. З іншого боку, інтеграція важка з іншими CRM.

Salesforce доступний за ціною, особливо якщо врахувати його широкий спектр можливостей. Навіть стартапи та малий бізнес можуть використовувати Salesforce.

Станом на травень 2016 року Salesforce мав понад 150 000 клієнтів по всьому світу. У світі CRM домінує Salesforce з часткою ринку 19,7%. Найближчі конкуренти - SAP (12,1%), Oracle (9,1%) та Microsoft (6,2%). В Salesforce AppExchange представлено понад 2700 додатків, що призвело до загальної кількості понад 3 мільйони встановлень, і понад 70% клієнтів Salesforce використовують програми, перелічені на AppExchange. Сьогодні багато компаній розробляють свої програми на платформі Salesforce або переходять на Salesforce. Це збільшило попит на розробників та адміністраторів Salesforce.

Компанія Salesforce розпочала свою діяльність як CRM-компанія, як Software as a Service (SaaS). Зараз Salesforce пропонує різні програмні рішення та платформу для користувачів та розробників для розробки та розповсюдження спеціального програмного забезпечення. Salesforce.com базується на архітектурі з декількома орендарями. Це означає, що декілька клієнтів мають спільну технологію, і всі вони працюють на останній версії. Не потрібно турбуватися про оновлення додатків чи інфраструктури - вони відбуваються автоматично. Це допомагає організації зосередитися на інноваціях, а не на управлінні технологіями.

Force.com, дозволяє адміністраторам та розробникам створювати веб-сайти та додатки в основному додатку Salesforce.com.

AppExchange - це інтернет-ринок додатків для сторонніх додатків, які працюють на платформі Force.com.

Heroku Enterprise надає розробникам гнучкість у створенні програм, використовуючи вибрані ними мови та інструменти.

Salesforce Thunder - це механізм обробки великих даних та правил, призначений для аналізу подій та здійснення персоніфікованих дій.

Пісочниця Salesforce дозволяє розробникам перевіряти ідеї в безпечному та ізольованому середовищі розробки.

Apex - це власна мова програмування, яку платформа Force.com надає розробникам, подібним до Java та C#. Це строго типізована об'єктно-орієнтована мова, нечутлива до регістру, мова програмування, дотримуючись крапкових позначень та синтаксису фігурних дужок. Apex можна використовувати для виконання запрограмованих функцій під час більшості процесів на платформі Force.com, включаючи користувацькі кнопки та посилання, обробники подій при вставці, оновленні чи видаленні записів, за допомогою планування або за допомогою користувацьких контролерів сторінок Visualforce або Lightning Experience. Використовує синтаксис, що виглядає як Java і діє як процедури, що зберігаються в базі даних, Apex дозволяє розробникам додавати бізнес-логіку до більшості системних подій, включаючи клацання кнопок, відповідні оновлення записів та сторінки Visualforce. Код Apex може ініціюватися за запитами веб-сервісу та від тригерів на об’єктах[10].

Як мова, Apex:

– Виклики мови обробки даних (DML), такі як INSERT, UPDATE і DELETE, які включають вбудовану обробку DmlException;

– Вбудовані мови запитів об’єктів Salesforce (SOQL) та мови пошуку об’єктів Salesforce (SOSL), які повертають списки записів sObject;

– Циклічна обробка, що дозволяє обробляти одночасно кілька записів;

– Блокування синтаксису, що запобігає конфліктам оновлення записів;

– Спеціальні загальнодоступні виклики API, які можна побудувати із збережених методів Apex;

– Попередження та помилки, що видаються, коли користувач намагається відредагувати або видалити спеціальний об'єкт або поле, на яке посилається Apex;

Apex базується на знайомих ідіомах Java, таких як синтаксис змінних та виразів, синтаксис блоків та умовних операторів, синтаксис циклу, нотація об’єктів та масивів. Там, де Apex вводить нові елементи, він використовує синтаксис та семантику, які легко зрозуміти та заохочують до ефективного

використання платформи Lightning. Тому Apex створює код, який є одночасно стислим і простим для написання[7].

Apex призначений для об’єднання декількох запитів та операторів DML в єдину одиницю роботи на сервері Salesforce. Розробники використовують сховані процедури баз даних, щоб аналогічно поєднати кілька операторів транзакцій на сервері баз даних. Як і інші процедури, що зберігаються в базі даних, Apex не намагається надати загальну підтримку рендерингу елементів в інтерфейсі користувача.

Apex - це строго типізована мова, яка використовує прямі посилання на об’єкти схеми, такі як імена об’єктів та полів. Він швидко зупиняє компіляцію, якщо будь-які посилання є недійсними. У ньому зберігаються всі власні залежності полів, об’єктів та класів у метаданих, щоб гарантувати, що вони не будуть видалені, якщо це вимагається активним кодом Apex.

Apex інтерпретується, виконується і повністю контролюється Lightning Platform.

Як і решта Lightning Platform, Apex працює в багатонаціональному середовищі. Отже, механізм виконання Apex розроблений для того, щоб ретельно захищати від втеченого коду, не даючи йому монополізувати спільні ресурси. Будь-який код, який порушує обмеження, блокується для розуміння повідомлень про помилки.

Apex надає вбудовану підтримку для створення та виконання модульного тесту. Він включає результати тесту, які вказують, скільки коду охоплено, і які частини вашого коду можуть бути більш ефективними. Salesforce гарантує, що всі власні коди Apex працюють належним чином, виконуючи всі модульні тести перед будь-якими оновленнями платформи.

Ви можете зберегти свій код Apex у різних версіях API. Це дозволяє вам зберігати поведінку.

Через багатокористувацьку природу платформи, мова суворо наклала обмеження губернатора для захисту від будь-якого коду, що монополізує спільні

ресурси. Salesforce пропонує серію асинхронних методів обробки для Apex, що дозволяє розробникам створювати більш тривалий та складніший код Apex.

Вбудовані програми Salesforce забезпечують потужну CRM-функціональність. Окрім того, Salesforce надає можливість налаштувати готові програми відповідно до вашої організації. Однак у вашій організації можуть бути складні бізнес-процеси, які не підтримуються наявною функціональністю. У цьому випадку Lightning Platform надає різні способи для просунутих адміністраторів та розробників створювати власні функціональні можливості.

Apex дозволяє:

– створення веб-служб;

– створення служб електронної пошти;

– виконайте складну перевірку декількох об'єктів;

– створюйте складні бізнес-процеси, які не підтримуються робочим процесом;

– створіть власну логіку транзакцій (логіку, яка відбувається протягом усієї транзакції, а не лише з одним записом або об’єктом);

– приєднайте власну логіку до іншої операції, наприклад, збереження запису, щоб вона відбувалася щоразу, коли виконується операція, незалежно від того, походить вона в користувальницькому інтерфейсі, на сторінці Visualforce чи з API SOAP.

Розробка компонентів Lightning, дозволяє налаштувати Lightning Experience, мобільний додаток Salesforce або створити власні самостійні програми. Ви також можете використовувати готові компоненти, щоб пришвидшити розробку.

Починаючи з весни 19 року (API версії 45.0), ви можете створювати компоненти Lightning, використовуючи дві моделі програмування: модель Lightning Web Components та оригінальну модель Aura Components. Веб-компоненти Lightning - це власні елементи HTML, побудовані з використанням HTML та сучасного JavaScript. Веб-компоненти Lightning та компоненти Aura можуть співіснувати та взаємодіяти на сторінці. Налаштуйте веб-компоненти

Lightning та компоненти Aura для роботи в Lightning App Builder та Experience Builder. Адміністратори та кінцеві користувачі не знають, яка модель програмування була використана для розробки компонентів. Для них це просто компоненти Lightning.

Visualforce складається з мови розмітки на основі тегів, що надає розробникам більш потужний спосіб створення додатків та налаштування інтерфейсу користувача Salesforce. За допомогою Visualforce можливо:

– Створення майстрів завантаження та інші багатоступеневі процеси;

– Створення власного контролю потоку через додаток;

– Визначення схеми навігації та правила, що стосуються даних, для оптимальної, ефективної взаємодії додатків.

У 2014 році Salesforce оприлюднила інтерфейс своєї платформи під назвою Lightning. На цій основній основі компонентів побудований мобільний додаток Salesforce. Salesforce побудував на цьому фреймворку в 2015 році, випустивши Lightning Design System, структуру стилю HTML із вбудованим стилем CSS за замовчуванням. Цей фреймворк дозволяє клієнтам створювати власні компоненти, які можна використовувати у своїх внутрішніх екземплярах або продавати на AppExchange.

Конструктор додатків Lightning Salesforce - це інструмент для швидкої розробки додатків адаптивних веб-інтерфейсів. Цей інтерфейс дозволяє створювати різні екрани на основі компонентів Lightning. Це можна використовувати як макети для записів або конкретних програм.

Lightning Experience, випущений у 2016 році, - це новий перероблений інтерфейс у Salesforce для вдосконалення процесів. Відтоді всі програми, доступні на AppExchange, повинні бути Lightning, а ті, що побудовані на Classic, повинні перейти на Lightning, оскільки Classic більше не буде оновлюватися Salesforce. Платформа пропонує можливість розробникам застосувати методи міграції, щоб увімкнути новий зручний інтерфейс і перейти на Lightning.

Увесь Apex працює повністю на замовлення на платформі Lightning. Розробники пишуть і зберігають код Apex на платформі, а кінцеві користувачі запускають виконання коду Apex через інтерфейс користувача (рис. 2.1.).

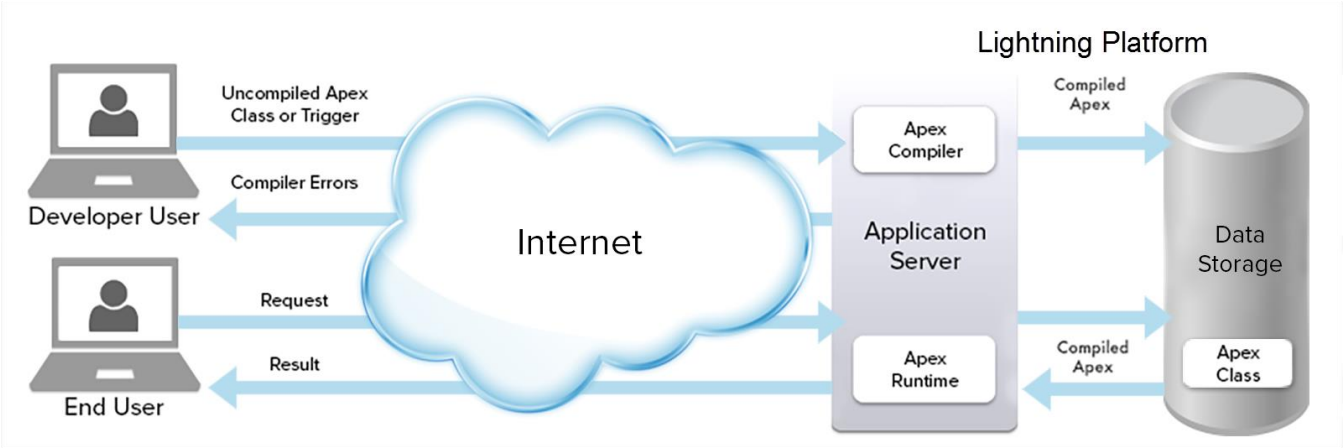

Рис. 2.1. Іллюструє як праціє APEX та Salesforce, як платформа

Коли розробник пише та зберігає код Apex на платформі, сервер додатків платформи спочатку компілює код в абстрактний набір інструкцій, зрозумілих інтерпретатору часу виконання Apex, а потім зберігає ці інструкції як метадані.

У Salesforce об’єкти поділяються на дві категорії, одна відома як стандартний об’єкт, а інша - як кастомний об’єкт. Стандартні об’єкти - це об’єкти, які Salesforce створює за замовчуванням, а кастомні об’єкти - це те, що Salesforce професійно розробляє відповідно до вимог замовника.

Подивіться на аркуш Excel нижче і майте на увазі візуальну концепцію. Тут вкладку можна розглядати як об'єкт. Рядки аркуша можна вважати записом, а заголовок стовпця - полем. Отже, з аркуша Excel доводиться, що об’єкт Salesforce - це накопичення полів (рис. 2.2).

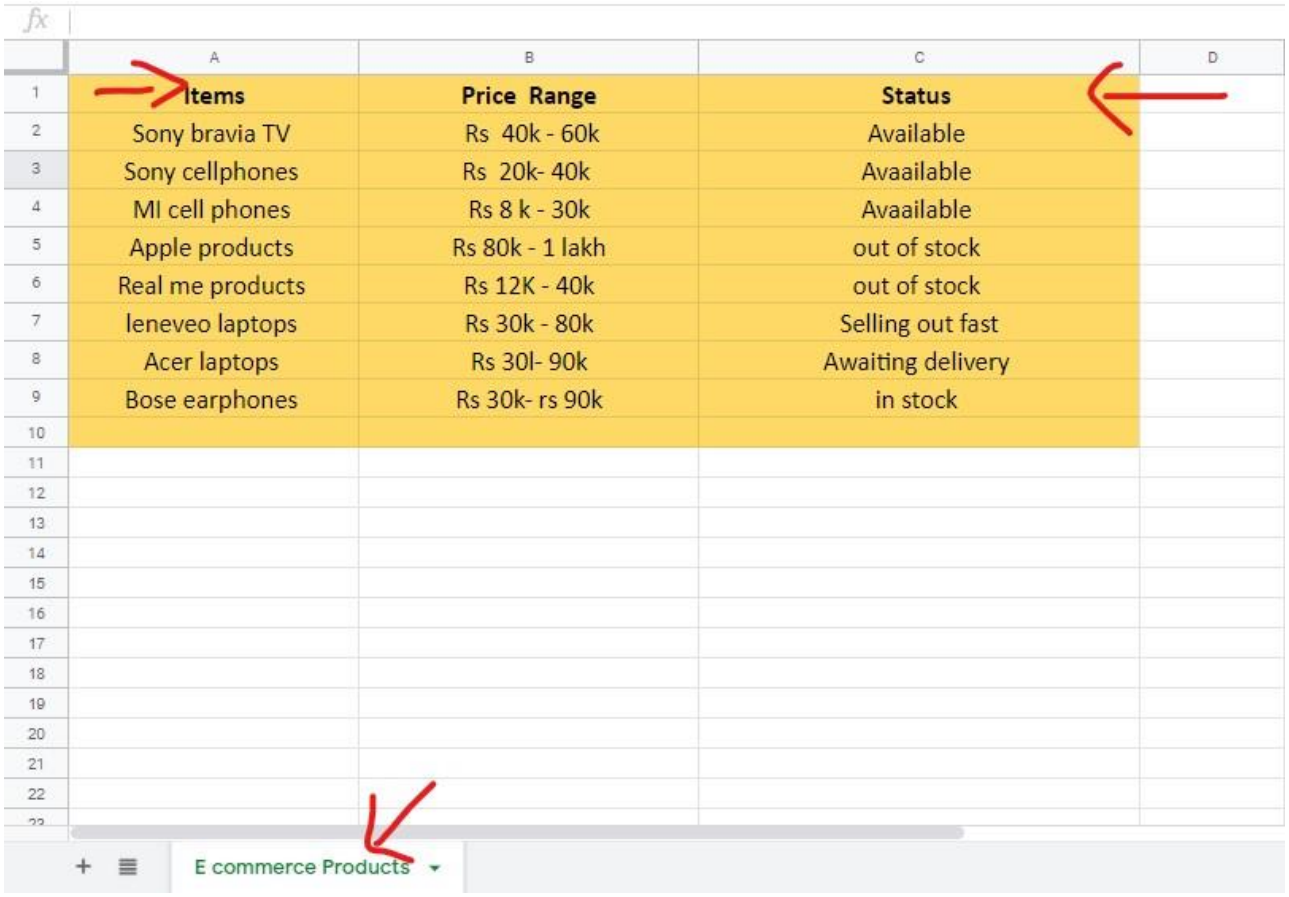

Рис. 2.2. Іллюстрація прикладу об'єкту

З наведеного зображення чітко видно, що вкладка електронної комерції розглядається як об’єкт Salesforce. Рядки розглядаються як записи і, нарешті, заголовок стовпця - як статус, діапазон цін, а товар - як поле.

Тепер обговоримо, що означає Salesforce під користувацьким об’єктом та стандартним об’єктом. Щоб у вас було чітке розуміння. Інформація про клієнта, як-от контакт, ім’я, адреса замовника, створюється за замовчуванням у Salesforce, отже, його можна назвати стандартним об’єктом. Подібним чином, коли ви хочете зберегти важливу ділову потребу замовника, він називається спеціальним об'єктом.

Приклад: власник бізнесу хоче відкрити веб-сайт електронної комерції. Тепер, щоб веб-сайт електронної комерції працював успішно, вам потрібна програма, яка зберігатиме деталі транзакцій клієнта, як номер картки та спосіб їх оплати, як за допомогою кредитної картки, дебетової картки або за допомогою прямого банківського переказу. Отже, професіоналу Salesforce потрібно

створити власний об’єкт для зберігання цієї інформації. Тепер у полі нестандартного об'єкта міститься номер картки клієнта, ім'я користувача, історія покупок, баланс гаманця та спосіб оплати.

Коли кінцевий користувач ініціює виконання Apex, можливо, натиснувши кнопку або перейшовши на сторінку Visualforce, сервер додатків платформи отримує складені інструкції з метаданих і надсилає їх через інтерпретатор середовища перед поверненням результату. Кінцевий користувач не спостерігає відмінностей у часі виконання від стандартних запитів платформи.

Salesforce оперує концепцією організацій або оргів. Як визначено Salesforce, організація:

"Розгортання Salesforce з визначеним набором ліцензованих користувачів. Організація - це віртуальний простір, що надається окремому клієнту Salesforce. Ваша організація включає всі ваші дані та програми та відокремлена від усіх інших організацій."

Коли клієнт купує Salesforce, йому надається організація, яка діє як контейнер для всіх їх даних.

Види орг

Існує дві широкі категорії організації:

- Екземпляри виробництва
- Екземпляри розробки

Організації виробництва призначені для використання з актуальними даними клієнтів та для активного ведення вашого бізнесу. Вони випускаються у 4 різних виданнях, у кожному з яких увімкнено різні функції:

- Essentials.
- Professional.
- Enterprise.
- Unlimited.

Екземпляри розробки використовуються для створення та тестування нових функцій та налаштувань перед тим, як їх випустити у виробничий екземпляр.

Існує кілька типів екземпляра розробки, кожен із яких має різні цілі та має різні набори функцій та можливостей налаштування. Найпоширенішими є:

- Повна копія пісочниці.
- Часткова копія пісочниці.
- Професійна для розробників.
- Видання розробника орг.
- Скретч орг.

Загалом, пісочниці використовуються для тестування функцій та отримання зворотного зв'язку з користувачами, тоді як організації розробників та подряпини використовуються як ізольоване середовище для роботи розробників.

Повна копія пісочниці = це повна копія виробничої організації, що включає дані та налаштування (метадані). Він найчастіше використовується для передвипускного тестування кінцевими користувачами.

Часткова копія пісочниці – це копіює налаштування та зразок даних виробничої організації (до 5 Гб). Він найчастіше використовується для інтеграції та тестування прийняття користувачами командами розробників.

Професійна для розробників - це копіює налаштування, але не виробничі дані. Організатори розробників можуть зберігати невелику кількість даних, які зазвичай використовуються для тестування. У цих організаціях розробники та адміністратори Salesforce будуватимуть і тестуватимуть нові функції.

Організація розробника. Середовище для розробників - це вільне середовище з меншим обсягом пам’яті, ніж організація розробника. Користувачі можуть зареєструватися на стільки, скільки їм потрібно. Організації видань розробників починаються як "порожні" шаблони і часто використовуються як початкове середовище розробки для невеликих робіт разом із організаціями розробників.

Скретч орга – Скретч-організацію нещодавно представив Salesforce як спосіб модернізації процесу розробки на платформі.

Скретч-організації - це короткочасні, одноразові та керовані джерелом середовища, що використовуються для швидкого тестування та розробки. Їх можна створювати на льоту за допомогою інструменту командного рядка та автоматично видаляти через короткий час. Вони, як правило, завантажуються дуже обмеженою підмножиною налаштувань та даних на певних етапах тестування, а потім видаляються.

2.4. Опис структури системи та алгоритмів її функціонування

Система витримана у вигляді MVC. У якості моделей виступають об’єкти Salesforce, за відображення відповідають visualforce-сторінки(VF-page), а у якості контролерів – класи APEX.

Шлях корчистувача до виконання сплати схематично виглядає наступним чином (рис. 2.3):

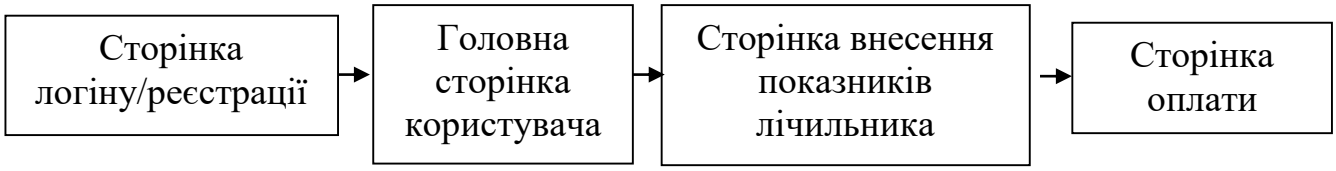

Рис. 2.3. Схема загального шляху користувача до етапу сплати

Під час проектування було проведно аналіз структури вхідних та вихідних даних, призначення веб-сервісуц, потоків інформації, складу користувачів та принципів їх роботи проектних рішень з урахуванням спрощення роботи користувачів.

Схема моделі даних виглядає наступним чином (рис. 2.4.):

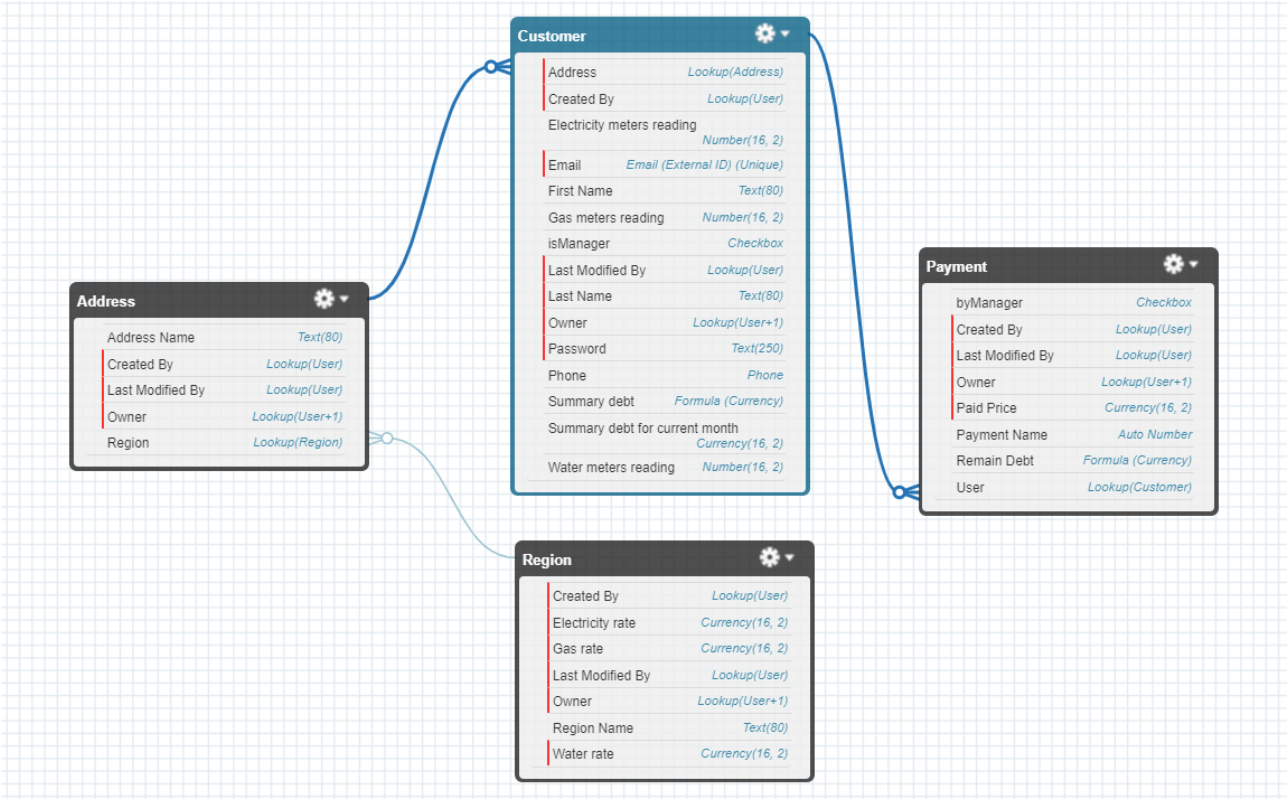

Рис. 2.4. Схема моделі даних

Структура веб-сервісу – це описання маршруту користувача, за яким він мандрує серед інтерфейсу. Чим простіше розроблений цей шлях – тим більше буде зацікавленість клієнта. Також дуже зручно, коли з будь-якого місця ти можеш повернутися назад.

Загальна структура веб-сервісу «Easy To Pay» з точки зору користувача (рис. 2.5.) має такий вигляд:

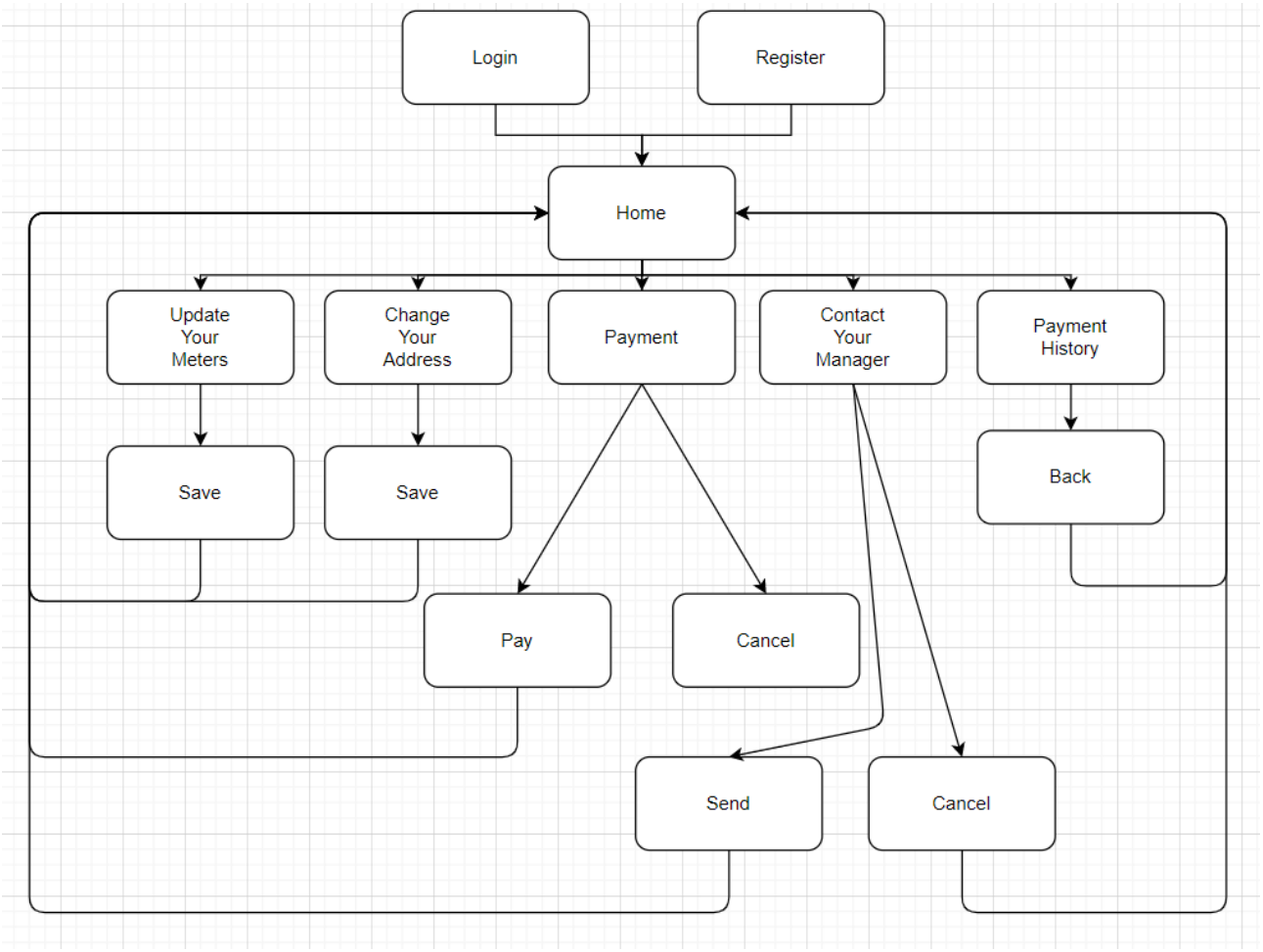

Рис. 2.5. Структура веб-сервісу «Easy To Pay»

Структура інтернет-магазину складається з таких частин:

- сторінка реєстрації користувача;
- головна сторінка з тарифами;
- введення нових показників лічильників;
- зміна адреси;
- сплата;
- зв'язок з менеджером;
- історія сплат.

Адміністративна панель складається з таких вкладок:

- список регіонів;
- інформація про користувачів;
- ролі користувачів;

– список адрес;
– усі виконані сплати.

При створенні сучасних веб-сервісів, останнім часом, використовують найпопулярніші шаблони проектування для створення архітектури додатку, такі як, наприклад, MVC і MVP.

Model-View-Controller (MVC, «Модель-Представлення-Контролер», «Модель-Вид-Контролер») - схема поділу даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, представлення і контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно. Він вирішує проблему поділу логіки обробки запиту користувача і логіки подання інформації (рис. 2.6.).

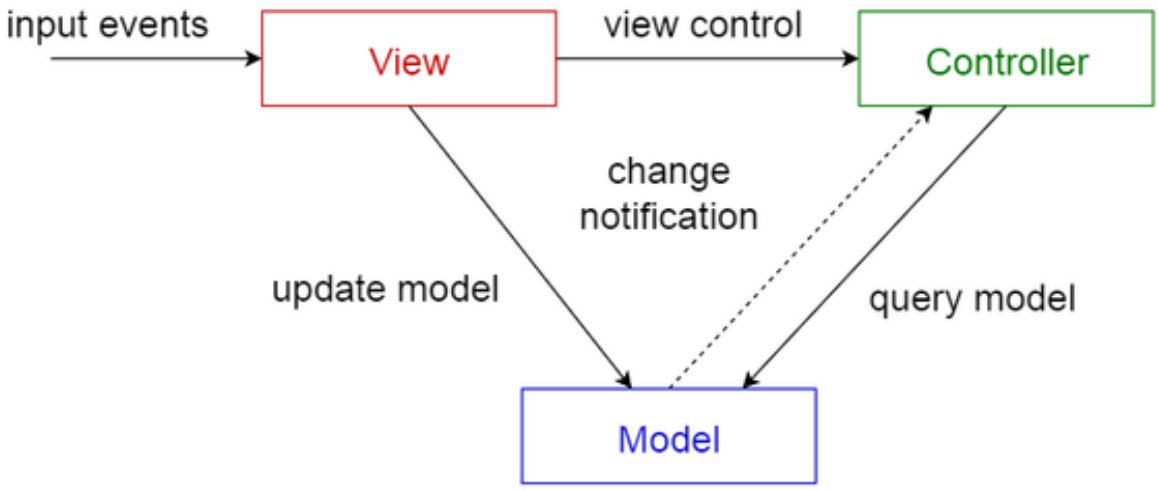

Рис. 2.6. Схема взаємодії компонентів веб-додатку в моделі MVC

Грамотна файлова система також має велике значення, бо якщо усі файли упорядковані та правильно підключені один до одного – усе працюватиме.

Коли ви працюєте на веб-сервісі локально на вашому комп'ютері, треба тримати все пов'язані файли в одній папці, яка відображає файлову структуру опублікованого веб-сервісі на сервері. Ця папка може розташовуватися де завгодно, але там, де можна легко її знайти, може бути, на ваш робочий стіл, в домашню папку або в корінь вашого жорсткого диска.

Також добре тримати файли у приватному репозиторії та кожну невелику частину нового коду комітити, щоб можна було легко повернутися до попередніх змін та завантажити код на комп’ютер при його втраті.

Файли веб-сервісу «Easy To Pay» зображені на рис. 2.6.

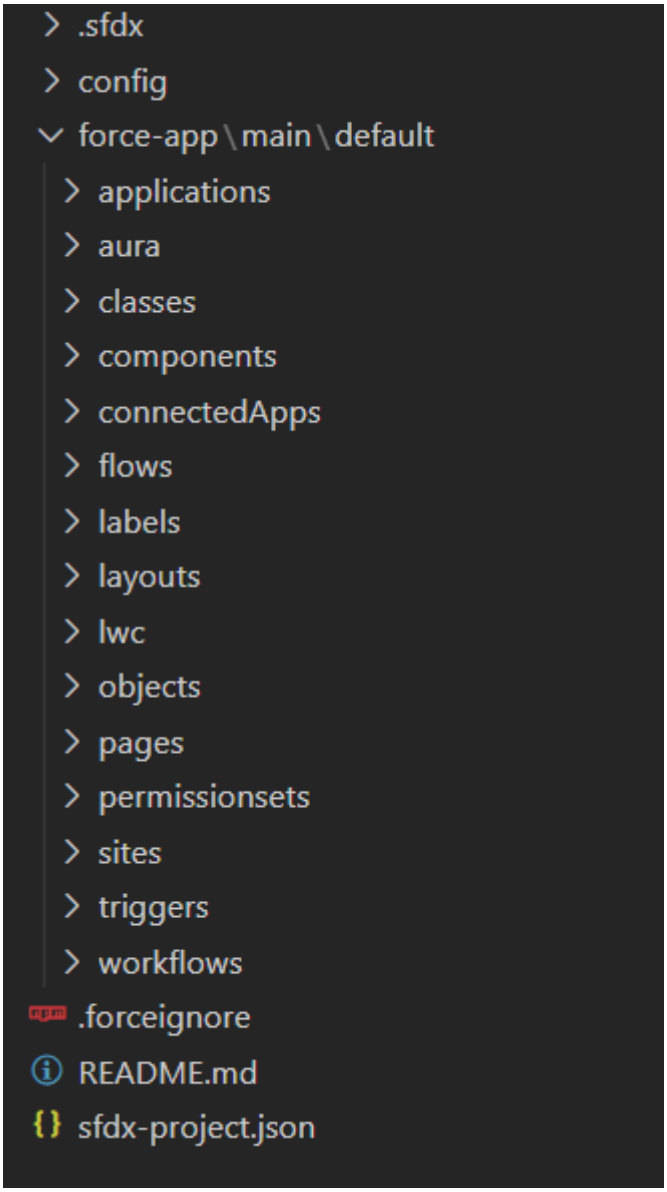

Рис. 2.6. Структура файлової системи

2.5. Обгрунтування та організація вхідних та вихідних даних програми

Програмне забезпечення отримує вхідні дані вручну.

Вхідні дані:

– тарифи на комунальні послуги;

- розмір сплати;
- звертання до менеджера;
- список адрес;
- список регіонів.

Вихідні дані:

- розмір заборгованності;
- квітанція про сплату;
- тарифи на комунальні послуги.

Дані програми організовані в локальні сховища даних, для кожної сторінки. Локальні сховища організують між собою одне велике сховище даних, до якого підключені усі основні компоненти веб-сервісу. Компоненти отримують дані з глобального сховища, та реагують на кожну зміну даних. Дані зі сховищ потрапляють за допомогою контролерів Salesforce.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

- Технічні засоби для працівання платформи Salesforce – це секретна інформація, яку коомпанія не надає, з доступної інформації відомо, що створена організація зберігається у двох географічно рознесених по світу серверах для дублювання оргінізації у випадку надзвичайної ситуації, у разі якої основний сервер буде вимкнено. Для, власне програмування, використовувався ноутбук з наступними характеристиками:
- марка – Acer;
- модель – Aspire 5 A515-56;
- процесор – Intel Core i3-8145;
- кількість ядер – 2 ядра;
- частота процесора – 2.3Ггц;
- оперативна пам’ять – 8 ГБ;

– тип пам’яті – DDR4;
– об’єм накопичувача – 256Гб;
– тип накопичувача – SSD;
– маніпулятор "миша";
– клавіатура.

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

2.6.2. Використані програмні засоби

Проект реалізований на мові програмування Apex з використанням Visualforce на базі Salesforce.

Розробка проводилась на операційній системі Windows 10.

Windows 10 - це основна версія Windows NT, розроблена корпорацією Майкрософт. Він замінить Windows 8.1, випущений майже два роки тому, і вийде 15 липня 2015 року та випущений 29 липня 2015 року. Windows 10 доступний для завантаження через MSDN та Technet як безкоштовне оновлення до Windows 8 та Windows 8 продуктів. Windows 8.1 через Магазин Windows і користувачі Windows 7 через Windows Update. Windows 10 постійно вітає нові будівлі, які можуть бути доступними для користувачів без додаткових витрат, а також різні версії тестування Windows 10, доступні людям з Windows. Частини ділового середовища можуть отримувати цю інформацію повільніше або використовувати більш тривалі періоди часу для отримання лише основних оновлень, таких як оновлення безпеки, протягом наступних 10 років життя.

У своєму першому випуску Windows 10 отримала переважно позитивні відгуки. Критики високо оцінили рішення корпорації Майкрософт надати настільні версії попередніх версій Windows на основі Windows, на відміну від

настільних ПК на базі Windows 8, хоча підхід, що базується на Інтернеті, дорікали за зворотне відстеження попередніх версій Touch. Критики також високо оцінили розробку сумісного з Windows 10 програмного забезпечення для Windows 8.1, інтеграцію Xbox Live, а також продуктивність та функції спеціального помічника Cortana та заміну Internet Explorer та Microsoft Edge. Однак ЗМІ критикували зміни поведінки в операційній системі, включаючи прийняття запропонованого закону, проблеми конфіденційності, пов'язані зі збором інформації про операційну систему для Microsoft та її партнерів, а також рекламне програмне забезпечення, подібне до використовуваної операційної системи.

Спочатку Microsoft намагався розмістити Windows 10 на більш ніж мільярді пристроїв протягом трьох років після її випуску; Цієї мети було нарешті досягнуто майже через п’ять років після випуску 16 березня 2020 року. У січні 2018 року Windows 10 перевершила Windows 7 як найпопулярнішу версію Windows у світі. За станом на червень 2021 року, за підрахунками, 79% комп’ютерів Windows, 58% усіх комп’ютерів (решта - це старша версія Windows та інших операційних систем, таких як macOS та Linux) та 24% усіх пристроїв Windows 10.

Visual Studio Code — засіб для створення, редагування та зневадження сучасних вебзастосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X.

Компанія Microsoft представила Visual Studio Code у квітні 2015 на конференції Build 2015. Це середовище розробки стало першим кросплатформовим продуктом у лінійці Visual Studio.

За основу для Visual Studio Code використовуються напрацювання вільного проєкту Atom, що розвивається компанією GitHub. Зокрема, Visual Studio Code є надбудовою над Atom Shell, що використовує браузерний рушій Chromium і Node.js. Примітно, що про використання напрацювань вільного проєкту Atom і на сайті Visual Studio Code, і в пресрелізі, і в офіційному блозі не згадується.

Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціюється як легковагове рішення, що дозволяє обійтися без повного інтегрованого середовища розробки. Серед підтримуваних мов і технологій: JavaScript, C++, C#, TypeScript, jade, PHP, Python, XML, Batch, F#, DockerFile, Coffee Script, Java, Salesforce, HandleBars, R, Objective-C, PowerShell, Luna, Visual Basic, Markdown, JSON, HTML, CSS, LESS і SASS, Haxe.

2.6.3. Виклик та завантаження програми

Для виклику та завантаженяя необхідно виконаті наступні дії:

– необхідно перейти за наступним посиланням: https://payyourservice-developer-edition.eu29.force.com/;
– зареєструватись на сайті;

2.6.4. Опис інтерфейсу користувача

Користувач може зареєструвати себе на сайті, вказавши свої особисті дані. Це можливо зробити на сторінці реєстрації (рис.2.7.).

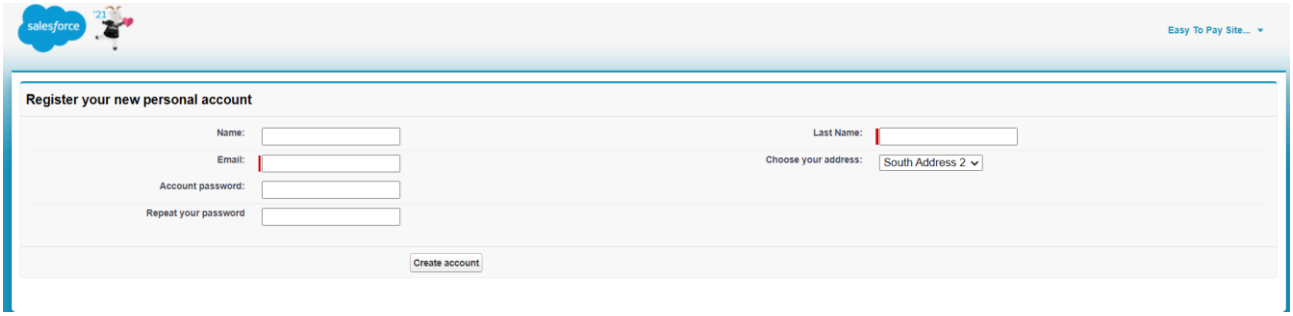

Рис. 2.7. Сторінка реєстрації

Після реєстрації клієнт може побачити основну сторінку (рис.2.8.), коли переходить на адресу сайту. На основній сторінці користувач бачить навігаційну панель у вигляді денкількох кнопок.

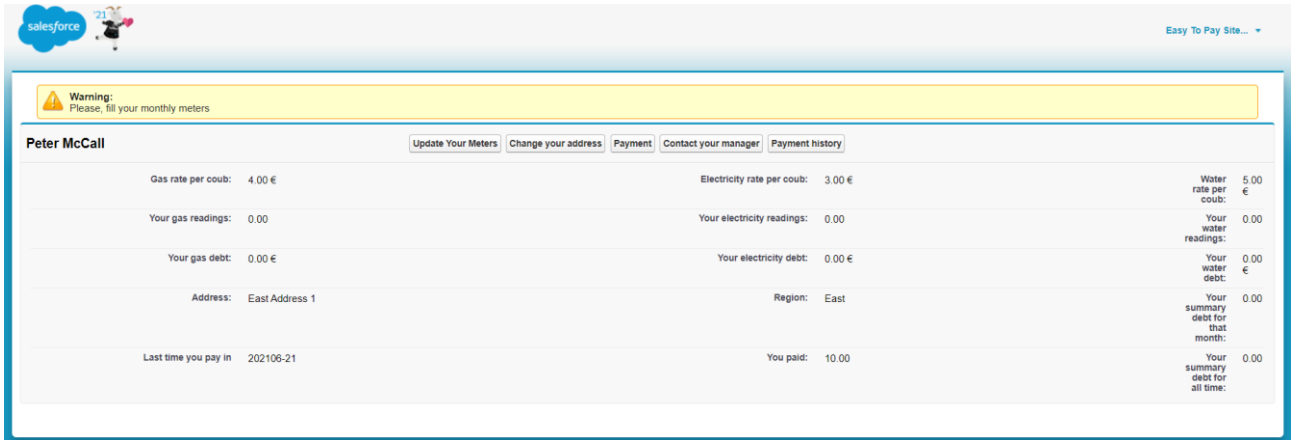

Рис. 2.8. Головна сторінка

Якщо користувач натисне кнопку «Update Your Meters» (рис.2.9) користувач потрапить до сторінки внесення ріхниці показників лічильника.

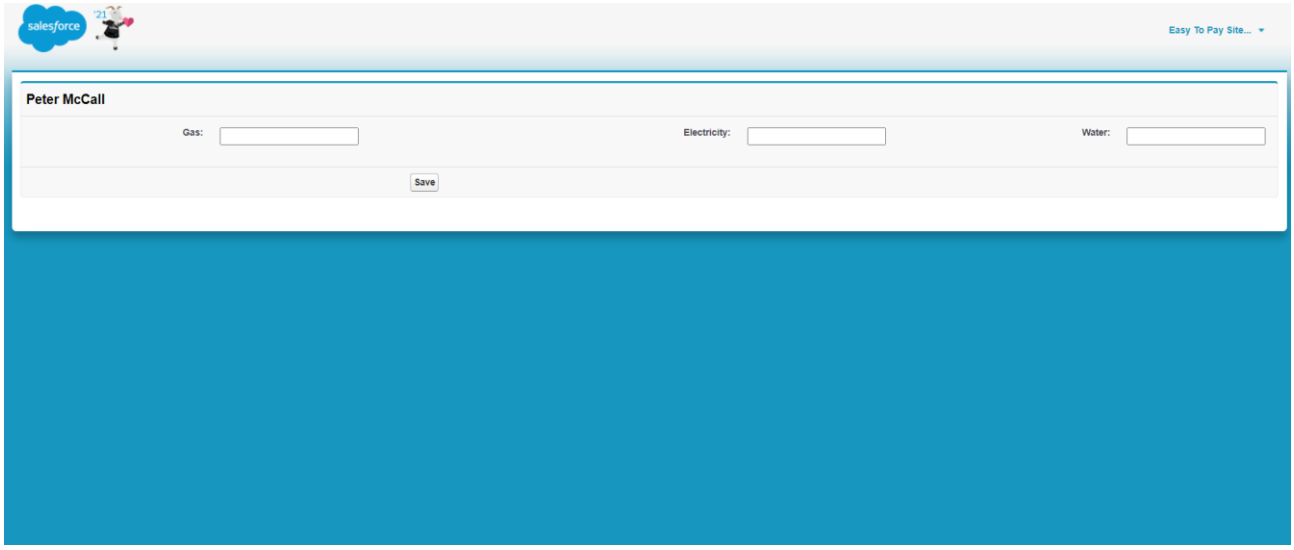

Рис. 2.9. Внесення показіників лічильника

Якщо користувач з головної сторінки натисне на кнопку «Change your address» він потрапить на сторінку зміни адреси (рис.2.10.). Там зі списку можна обрати нову адресу мешкання.

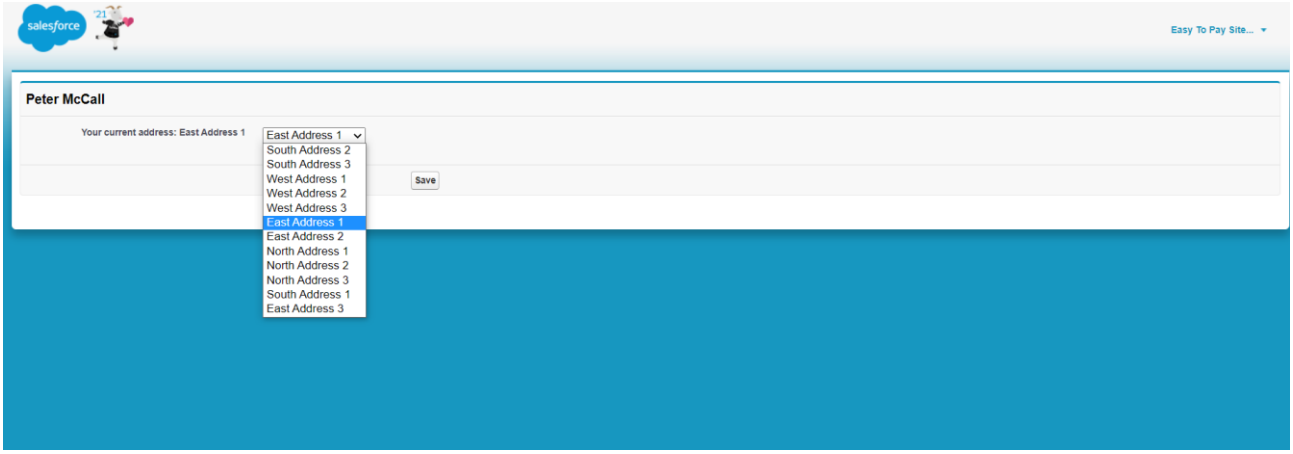

Рис. 2.10. Сторінка зміни адреси

Коли користувач переходить у вкладку «Payment» (рис.2.11.), він потрапить на сторінку сплати, де можна буде внести необхідний розмір сплати. Система налаштована таким чином, що сплатити можна частково, повністю та наперед із запасом для наступних сплат. Після сплати на пошту буде відправлена квітанція про сплату.

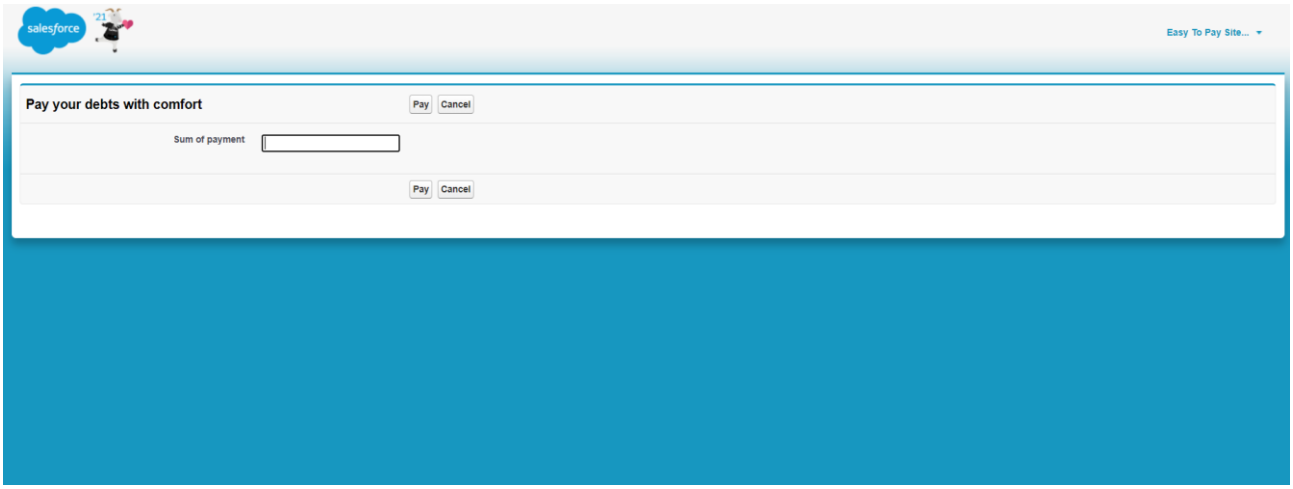

Рис. 2.11. Сторінка сплати

Якщо користувач натисне на кнопку «Contact your manager», то потрапить на сторінку зворотьного зв’язку з менеджером (рис.2.12.).

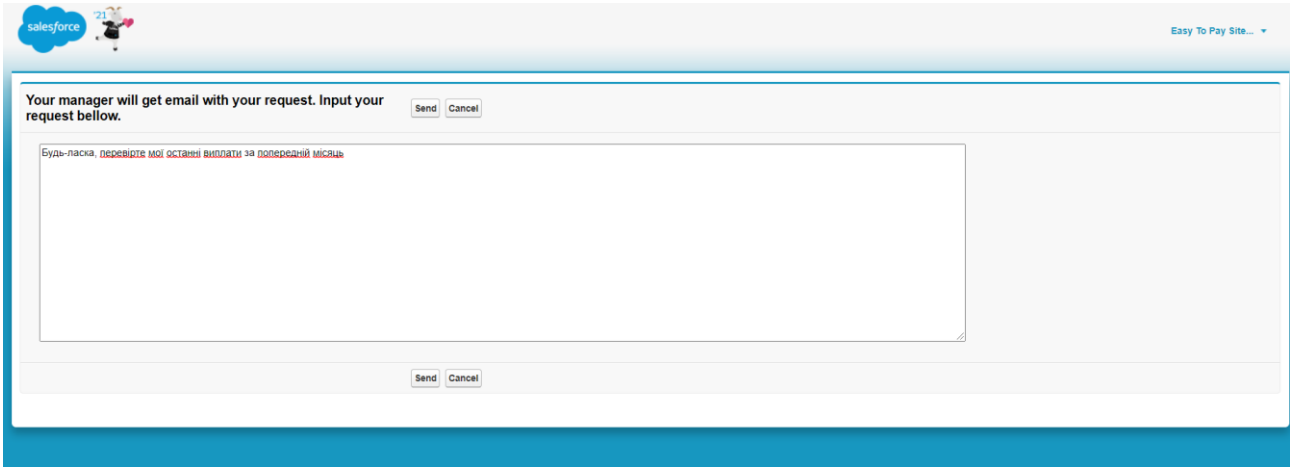

Рис. 2.12. Сторінка зворотнього зв’язку

Якщо користувач натисне на кнопку «Payment history», то потрапить на сторінку, де можна перевірити свої останні оплати (рис.2.13.)

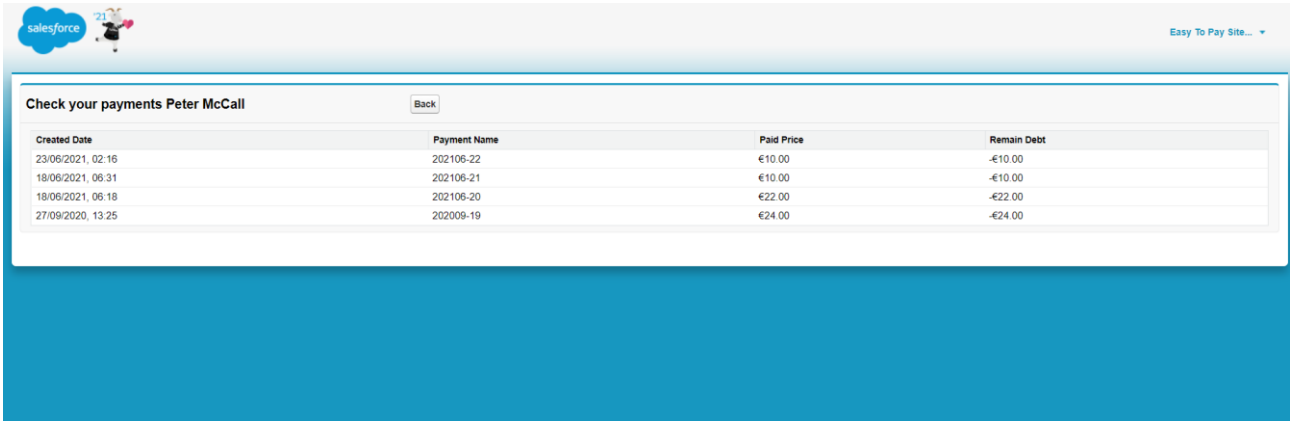

Рис. 2.13. Сторінка історії оплат

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 3050;
2. коефіцієнт корекції програми в ході її розробки – 0,05;
3. коефіцієнт складності програми – 1,7;
4. годинна заробітна плата програміста– 125 грн/год;

Середня годинна зарплата Junior Salesforce developer в Україні була вирахувати виходячи з даних «Української спільноти програмістів (DOU)» [10]. Станом на кінець 2020 року зарплата Junior Salesforce розробника простягається від 500$ до 1100$. Вирахувавши середню заробітну плату програміста маємо плату 800$ у місяць. При курсі валют НБУ на початок червня 2021 року один американський долар дорівнює 27,34 грн, тому середня зарплата в гривнях дорівнює 21920 грн. При стандартному графіку (176 годин/місяць) зарплата за годину буде становити близько 125 грн.

5. коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі – 1,4;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,4;
7. вартість машино-години ЕОМ –21 грн/год.

Оскільки для цього проекту протрібна велика потужність ПК для бекенду та підняття великої кількості даних на локальному сервері, гарним рішенням буде аренда комп’ютера на час розробки додатку. Вартість аренди комп’ютера на місяць 1300 грн (монітор) та 2400 грн (системний блок). Загалом на місяць оренда коштуватиме 3700 грн. При стандартному графіку (176 годин/місяць) вартість машино-години ЕОМ за годину роботи буде становити 21 грн. В цю

вартість входить ремонт за гарантією та базовий комплект гарнітури (клавіатура та миша) [9].

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \quad (3.1)$$

де t_o- витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a- витрати праці на розробку блок-схеми алгоритму;

t_n-витрати праці на програмування по готовій блок-схемі;

$t_{отл}$-витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмногму забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де q - передбачуване число операторів (3050);

C - коефіцієнт складності програми (1,7);

p - коефіцієнт корекції програми в ході її розробки (0,05).

Звідси умовне число операторів в програмі:

$$Q = 1{,}7 \cdot 3050 \cdot (1 + 0{,}05) = 5444{,}25$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k} \text{ , людино-годин,}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 5 до 8 років він складає 1,4.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1,2). З урахуванням коефіцієнта кваліфікації k = 1,4, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (5444{,}25 \cdot 1{,}2) / (75 \cdot 1{,}4) = 62{,}22 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25) \cdot k} \text{ , людино-годин, (3.2)}$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.2), отримаємо:

$$t_a = 5444{,}25 / (20 \cdot 1{,}4) = 194{,}4 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}$$, людино-годин.

t_n = 5444,25 / (25 · 1,4) = 155,55 людино-годин.

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}$$, людино-годин.

$t_{отл}$= 5444,25 / (5 · 1,4) = 777,75 чел.-ч.

- за умови комплексного налагодження завдання:

$$t^k_{отл} = 1{,}5 \cdot t_{отл}$$, людино-годин.

$t^k_{отл}$= 1,5 · 777,75 = 1166,6 людино-годин.

Витрати праці на підготовку документації визначаються за формулою:

$$t_д = t_{др} + t_{до}$$, людино-годин,

де $t_{др}$-трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20)\cdot k}$$, людино-годин,

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др}$$, людино-годин.

Підставляючи відповідні значення, отримаємо:

$t_{др}$ = 5444,25 / (18 · 1,4) = 245,2 людино-годин.

$t_{до}$ = 0,75 · 245,2 = 183,9 людино-годин.

$t_{д}$ = 245,2 + 183,9 = 429,1 людино-годин.

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

t = 50 + 62,22 + 194,4 + 155,55 + 777,75 + 429,1 = 1669 людино-годин.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}$$, грн.

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}$$, грн,

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 125 грн / год, отримуємо:

$$З_{ЗП} = 1669 \cdot 125 = 208\ 625 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot C_{мч}, \text{ грн,} \quad (3.3)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ - вартість машино-години ЕОМ, грн/год (21 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 777{,}75 \cdot 21 = 16\ 332{,}75 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 208\ 625 + 16\ 332{,}75 = 224\ 957{,}75 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.}$$

де B_k- число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси витрати на створення програмного продукту:

$$T = 1669 / 1 \cdot 176 \approx 9{,}48 \text{ міс.}$$

Висновок: програмне забезпечення розроблено для забезпечення доступу користувачів до основних тарифів комунальних послуг, сплати заборгованності згідно лічильників та обліку надання відповідних послуг. Вартість даного програмного забезпечення близько 224 957,75 тис. грн і не вимагає додаткових витрат як при розробці програми. Очікуваний час розробки становить 1669 годин, тобто 9,48 місяці. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було поставлено завдання розробити веб-орієнтованої інформаційної системи обліку надання комунальних послуг з використанням Salesforce.

Це програмне забезпечення призначене для надання можливості сплачувати та вести облік надання комунальних послуг. Практичне призначення даної системи полягає в забезпеченні зручної та універсальної системи для сплати комунальних сервісів.

Під час виконання даного проекту були виконані наступні задачі:

- вивчено предметну галузь розв'язуваної задачі;
- створено алгоритм для реалізації поставленого завдання;
- створено базу даних.

Розроблене програмне забезпечення дозволяє:

- легке та доступне адміністрування веб-сервісу;
- формування квітанцій про сплату за комунальні послуги;
- доступ до актуальних тарифів постачальників комунальних послуг;
- контакування з менеджером регіону;
- сплата заборгованностей, згідно показникам лічильників.

Було розроблено веб-орієнтовану інформаційну систему обліку надання комунальних послуг з використанням Salesforce, як мова програмування була застосована APEX, фронтенд мова Visualforce, для керування проектом була використана програма Vs Code, а для керування версіями веб-сервісу було використано Git.

Це програмне забезпечення призначене для надання можливості сплачувати та вести облік надання комунальних послуг. Практичне призначення даної системи полягає в забезпеченні зручної та універсальності системи для сплати комунальних сервісів для людей які проживають за межами СНГ простору.

Також у кваліфікаційній роботі було визначено трудомісткість розробленої системи, на базі середьої зарплати розробника проведений підрахунок вартості

роботи по створенню програми, який складає 224 957,75 грн та розраховано час на створення інтернет-магазину - 1669 людино-годин , тобто 9,48 місяців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вебінар на тему веб-додатків та провідних веб-технологій [Електронний ресурс] - Режим доступу: https://dou.ua/calendar/37625/
2. Скотт Б. Проектирование веб-интерфейсов / Б. Скотт, Т. Нейл. —Спб.: Символ-Плюс, 2010. — 352 с.
3. Addison.Wesley.Force.com.3rd.Edition – C.: O'Reilly,2010. – 267 с.
4. Марк Тиленс Томас. Apex в действии. Питер, 2018. -368 с. ISBN - 5446109996, 9785446109999
5. Скотт Б. Проектирование веб-интерфейсов / Б. Скотт, Т. Нейл. —Спб.: Символ-Плюс, 2018 — 352 с.
6. Advanced Apex Programming for Salesforce.com and Force.com - Dan Appleman. -304 с. ISBN - 978-5-496-03003-8
7. Інькова Н. А. Створення Web-сайтів: Навчально-методичний посібник [Електронний ресурс] / Інькова Н.А., Зайцева Е.А., Кузьміна Н.В., Толстих С.Г. / / Режим доступу: http://club-edu.tambov.ru/methodic/fio/p5.doc
8. М. Хавербеке – Выразительный Visualforce. Современное веб-программирование 2018. –С. 125-329.
9. Вартість аренди ноутбуку почасово [Електронний ресурс] - Режим доступу: https://notebooksbu.com/garantiya-3-goda/
10. Середня заробітна плата Junior Salesforce developer в Україні станом на початок 2021 року. https://dou.ua/lenta/articles/salary-report-devs-june-2020/
11. Data Management [Електронний ресурс] – Режим доступу: https://trailhead.salesforce.com/content/learn/modules/lex_implementation_data_management?trail_id=force_com_admin_beginner
12. Platform Development Basics [Електронний ресурс] – Режим доступу: https://trailhead.salesforce.com/content/learn/modules/platform_dev_basics?trail_id=force_com_dev_beginner

13. Молер, Дж. Flash 8. Руководство Web-дизайнера; Эксмо - М., 2019. - 400 с.

14.Веб-приложение и его виды [Електронний ресурс] - Режим доступу: https://semantica.in/blog/veb-prilozhenie.html

15.Website и Web Application: в чем разница? [Електронний ресурс] - Режим доступу: https://dinarys.com/ru/blog/websitevs.-webapplication

16. Разработка мобильных и веб-приложений: что является лучшим решением [Електронний ресурс] - Режим доступу: https://smartum.pro/ru/blog-ru/razrabotka-mobilnykh-i-webprilozheniy/

17.Орлов Л. А. Як створити електронний магазин в Інтернет / Л. А. Орлов. – М.: БУК-ПРЕС, 2016. – 384 с

18.Стотлемайер Д. Тестирование Web-приложений / Д. Стотлемайер – М.: Кудиц-образ, 20017. – 240 с.

19. Data Modeling [Електронний ресурс] – Режим доступу: https://trailhead.salesforce.com/content/learn/modules/data_modeling?trail_id=force_com_admin_beginner

20. Salesforce platform basic [Електронний ресурс] – Режим доступу: https://trailhead.salesforce.com/content/learn/modules/starting_force_com?trail_id=force_com_admin_beginner

21.Пасічник Н.Р. Формалізм в постановці задачі створення якісного сайту / Н.Р. Пасічник // Наукові праці Донецького національного технічного університету. Інформатика, кібернетика та обчислювальна техніка. – Донецьк. – 2017. – Вип. 14 (188). – С. 325-329.

22.Горнаков, С.Г. Осваиваем популярные системы управления сайтом / Горнаков С.Г. – ДМК Пресс, 2019 – С. 336.

23.Савельева Н. Системы управления контентом / Савельева Н. // Открытые системы, 2020. — С. 41-47.

24.Комисаров Д.А., Станкевич А.Г. Персональный учитель по персональному компьютеру. – М, 2020.

25.Леонтьєв В.П. «Web-дизайн. Керівництво користувача »

26.Гукин Д. - FrontPage для "чайников". - К.,2016.- С. 102-123.

27. Salesforce Admin. [Електронний ресурс] – Режим доступу: https://trailhead.salesforce.com/content/learn/trails/force_com_admin_beginner

28.Э. Браун – Изучаем Visualforce. Руководство по созданию современных веб-сайтов 2019. — С. 35-47

29. Salesforce Classic [Електронний ресурс] - Режим доступу: https://trailhead.salesforce.com/content/learn/modules/admin_intro_accounts_contacts

30. Основы Apex [Електронний ресурс] - Режим доступу: https://trailhead.salesforce.com/content/learn/modules/admin_intro_crm_basics

ДОДАТОК А

КОД ПРОГРАМИ

```
public class AllUsersPaymentHistoryController{

    public List<Customer__c> regionUsers{get;set;}
    public List<Payment__c> payments{get;set;}
    public Customer__c selectedUser{get;set;}
    public String userId{get;set;}

    public AllUsersPaymentHistoryController(){
        String regionId = PageReferenceCreator.getId();
        regionUsers = [SELECT Name, Last_Name__c, Summary_debt_for_current_month__c,
                        (SELECT Name, Paid_Price__c, CreatedDate, Remain_debt__c, byManager__c
                         FROM Payments__r
                         ORDER BY CreatedDate DESC)
                       FROM Customer__c
                       WHERE Address__r.region__r.Id = :regionId
                       ORDER BY Name];

        userId = '';
        selectedUser = new Customer__c();
        payments = new List<Payment__c>();

    }

    public List<SelectOption> getUsers(){
        return GetListOfUsers.listOfUsers(regionUsers);
    }

    public PageReference rerender(){
        for (Customer__c user : regionUsers) {
            if(user.Id == userId){
                selectedUser = user;
            }
        }

        for (Payment__c payment : selectedUser.Payments__r) {
            payments.add(payment);
        }
        return null;
    }

    public PageReference back(){
        String managerId = PageReferenceCreator.getParam('managerid');
        return PageReferenceCreator.createReference(PathConst.MANAGERWORKFLOW, managerId);

    }

}

public class APIParams {
    public static final String CLIENTID =
'3MVG91BJr_0ZDQ4tV15NHF357DzlGSBl3Gu0fcthIPZbkjvXnb24sV7VIUGzdbzWgfalcMpxwXR.PdqLwvXHg';
    public static final String GRANTTYPE = 'password';
    public static final String CLIENTSECRET =
'7D71224CE7E2FC164BBD95A13B07C672A1A16A17EB28E2F30146EEAD7B3ABC56';
    public static final String USERNAME = 'oleksii.ilin@alexinc.com';
    public static final String PASSWORD = 'Apple2019';
    public static final String USERSECRET = 'pqAe6vpf5iLDtJ5GEQhP8zdR';
```

```
    public static final String LOGINENDPOINT = 'https://login.salesforce.com/services/oauth2/token';
    public static final String PAYMENTAPIENDPOINT =
'https://eu29.salesforce.com/services/apexrest/Payment__c/';
}

public class BillController {

    public List<Payment__c> bills{get;set;}
    public Payment__c bill{get;set;}
    public Datetime dateOfPayment{get;set;}

    public BillController(){
        String billId = Apexpages.currentPage().getParameters().get('id');
        bills = [SELECT Id, Name, CreatedDate, Paid_Price__c, Remain_debt__c,
                    User__r.Name, User__r.Last_Name__c, User__c
                FROM Payment__c
                WHERE Id=:billId
                LIMIT 1];
        if (bills.isEmpty() == false) {
            bill = bills.get(0);
        }

        dateOfPayment = bill.CreatedDate;

    }

}

public class ChangeAddressController{

    public Customer__c user{get; set;}

    public ChangeAddressController(){
        String userId = PageReferenceCreator.getId();
        user = [SELECT Id, Name, Last_Name__c, Address__c, Address__r.Name
                FROM Customer__c
                WHERE id = :userId
                LIMIT 1];

    }

    public List<SelectOption> getAddresses(){
        return GetListOfAddresses.getAddresses();
    }

    public PageReference save(){
        update user;
        return PageReferenceCreator.createReference(PathConst.USERWORKFLOW, user.Id);
    }

}

public class ChangeMonthlyPaymentController{

    public List<Customer__c> regionUsers{get;set;}
    public Customer__c selectedUser{get;set;}
    public String userId{get;set;}
    public String paidPrice{get;set;}

    public ChangeMonthlyPaymentController(){
        String regionId = PageReferenceCreator.getId();
        regionUsers = [SELECT Name, Last_Name__c, Summary_debt_for_current_month__c, Email__c
                    FROM Customer__c
```

```
                    WHERE Address__r.Region__r.Id = :regionId
                    ORDER BY Name];

        userId = '';
        paidPrice = '';
        selectedUser = new Customer__c();

    }

    public List<SelectOption> getUsers(){
        return GetListOfUsers.listOfUsers(regionUsers);
    }

    public PageReference back(){
        String managerId = PageReferenceCreator.getParam('managerid');
        return PageReferenceCreator.createReference(PathConst.MANAGERWORKFLOW, managerId);
    }

    public PageReference rerender(){
        for (Customer__c user : regionUsers) {
            if(user.Id == userId){
                selectedUser = user;
            }
        }
        return null;
    }

    public PageReference save(){
        if(paidPrice != ''){
            Map<String, String> params= new Map<String, String>();
            params.put('paidPrice', paidPrice);
            params.put('userId', selectedUser.Id);
            params.put('isManager', 'true');
            HttpResponse response = new HttpResponse();

            //send request to create new payment

            try {
                HttpResponse accessTokenResponse = PaymentControllerHelper.getAccessKey();
                Map<String,Object> result = (Map<String, Object>)
JSON.deserializeUntyped(accessTokenResponse.getbody());
                String accessToken =  String.valueOf(result.get('access_token'));
                response = PaymentControllerHelper.createPayment(params, accessToken);
            }
            catch (Exception ex) {
                MessageThrower.addMessageError(WarningConst.PAYMENTFAILED+'we here');
                return null;
            }

             //catching ok status
            if (response.getStatusCode() == 200) {
                //sending bill
                String paymentId =  (String)JSON.deserializeUntyped(response.getbody());
                PaymentControllerHelper.sendBill(PaymentControllerHelper.createSuccessBill(selectedUser,
paymentId));
                MessageThrower.addMessageConfirm(WarningConst.MONTHLYPAYMENTUPDATED);

                return null;

            } else {
                MessageThrower.addMessageError(WarningConst.PAYMENTFAILED+response.getStatusCode());
                PaymentControllerHelper.sendBill(PaymentControllerHelper.createFailedBill(selectedUser));
                return null;
```

```
            }

        }
        else {
            MessageThrower.addMessageWarning(WarningConst.EMPTYPRICE);
            return null;
        }

    }
}

public class ChangeRatesController{

    public Region__c region{get;set;}

    public ChangeRatesController(){
        String regionId = PageReferenceCreator.getId();

        region = [SELECT Name, Gas_rate__c, Electricity_rate__c, Water_rate__c
                FROM Region__c
                WHERE Id = :regionId
                LIMIT 1];
    }

    public PageReference save(){
        update region;
        MessageThrower.addMessageConfirm(WarningConst.RATESSAVED);
        return null;
    }

    public PageReference back(){
        String managerId = PageReferenceCreator.getParam('managerid');
        return PageReferenceCreator.createReference(PathConst.MANAGERWORKFLOW, managerId);
    }
}

public class ContactManagerController{

    public Customer__c user{get;set;}
    public Customer__c manager{get;set;}
    public String emailText{get;set;}

    public ContactManagerController(){
        String userId = PageReferenceCreator.getId();
        user = [SELECT Id, Name, Last_Name__c, Email__c
                FROM Customer__c
                WHERE id = :userId
                LIMIT 1];

        manager = [SELECT Id, Name, Last_Name__c, Email__c
                  FROM Customer__c
                  WHERE isManager__c = true
                  LIMIT 1];
        emailText='';

    }

    public PageReference sendToManager(){

        if(emailText != ''){
            Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
            email.toAddresses = new String[]{manager.Email__c};
            email.setSubject('Contact request from ' + user.Name + ' ' + user.Last_Name__c);
```

```
                email.optOutPolicy = 'FILTER';
                email.plainTextBody = manager.Name + ' ' + manager.Last_Name__c + ', ' + emailText + '\n You may
send your answer on user\'s email:\n' + user.Email__c;
                Messaging.SingleEmailMessage[] messages =   new List<Messaging.SingleEmailMessage> {email};
                Messaging.sendEmail(messages);

                return PageReferenceCreator.createReference(PathConst.USERWORKFLOW, user.Id);
            }
            else {
                MessageThrower.addMessageWarning(WarningConst.EMPTYEMAILTEXT);
                return null;

            }
        }

        public PageReference cancel(){
            return PageReferenceCreator.createReference(PathConst.USERWORKFLOW, user.Id);
        }
    }

    public class GetListOfUsers {
        public static List<SelectOption> listOfUsers(List<Customer__c> users){
            List<SelectOption> options = new List<SelectOption>();
            for(Customer__c user : users){
                options.add(new SelectOption(user.Id, user.Name+' '+user.Last_Name__c));
            }
            return options;
        }


    }

    public class LoginController{
      public Customer__c user {get;set;}
        public String login {get;set;}
        public String password {get;set;}

        public LoginController(){
            user = new Customer__c();
            login = '';
            password = '';
        }

        public PageReference getCustomer(){
            try {
                user = [SELECT Id, Email__c, Password__c, isManager__c
                        FROM Customer__c
                        WHERE Email__c = :login AND Password__c = :password
                        LIMIT 1];
            } catch (Exception e) {
                MessageThrower.addMessageError(WarningConst.LOGINERROR);
                return null;
            }

            return signIn();
        }

        public PageReference register(){
            return PageReferenceCreator.createReference(PathConst.REGISTER);
        }

        public PageReference signIn(){
            return PageReferenceCreator.createReference(PathConst.USERWORKFLOW, user.Id);
```

```
    }

    public PageReference managerWorkflow(){
        PageReference pageRef;

        try {
            user = [SELECT Id, Email__c, Password__c, isManager__c
                        FROM Customer__c
                        WHERE Email__c = :login AND Password__c = :password
                    LIMIT 1];

        } catch (Exception e) {
            MessageThrower.addMessageError(WarningConst.LOGINERROR);
        }

        if (user.isManager__c) {
            pageRef =  PageReferenceCreator.createReference(PathConst.MANAGERWORKFLOW, user.Id);
        }
        else{
            MessageThrower.addMessageError(WarningConst.PERMISSIONERROR);
        }

        return pageRef;

    }


}

public class ManagerWorkflowController{
    public Customer__c manager{get;set;}
    public Region__c region {get;set;}
    public Address__c address {get;set;}

    public ManagerWorkflowController(){
        String managerId = PageReferenceCreator.getId();
        try {
            manager = [SELECT Id, Name, Last_Name__c, Address__c, Address__r.Region__c,
Address__r.Region__r.Id
                        FROM Customer__c
                        WHERE Id = :managerId
                        LIMIT 1];
            address = [SELECT Region__c, Region__r.Id
                        FROM Address__c
                        WHERE Id = :manager.Address__c];

            region = [SELECT Name, Gas_rate__c, Electricity_rate__c, Water_rate__c
                    FROM Region__c
                    WHERE Id = :address.Region__r.Id
                    LIMIT 1];

        } catch (Exception e) {
            MessageThrower.addMessageError(e.getMessage());
        }


    }

    public PageReference changeRates(){
        System.debug(PathConst.CHANGERATES+' - page');
        System.debug(region + ' - region');
        System.debug(manager + ' - manager');
```

```
            return PageReferenceCreator.createReference(PathConst.CHANGERATES, region.Id, manager.Id);
        }

        public PageReference updateMonthlyPayment(){
            return PageReferenceCreator.createReference(PathConst.CHANGEPAYMENT, region.Id, manager.Id);
        }

        public PageReference usersPaymentHistory(){
            return PageReferenceCreator.createReference(PathConst.ALLUSERSHISTORY, region.Id, manager.Id);
        }

        public PageReference viewAllUsers(){
            return PageReferenceCreator.createReference(PathConst.ALLUSERS, region.Id, manager.Id);
        }
    }

    public class MessageThrower {
        public static void addMessageConfirm(String message){
            ApexPages.Message msg = new ApexPages.Message(ApexPages.Severity.CONFIRM, message);
            ApexPages.addMessage(msg);
        }
        public static void addMessageError(String message){
            ApexPages.Message msg = new ApexPages.Message(ApexPages.Severity.ERROR, message);
            ApexPages.addMessage(msg);
        }
        public static void addMessageWarning(String message){
            ApexPages.Message msg = new ApexPages.Message(ApexPages.Severity.WARNING, message);
            ApexPages.addMessage(msg);
        }
    }

    public class PageReferenceCreator{

        public static PageReference createReference(String url, String id){
            PageReference pageRef = new PageReference(url);
            pageRef.setCookies(new Cookie[]{new Cookie('id', id, '/', 300, true)});
            return pageRef;
        }

        public static PageReference createReference(String url, String id, String managerId){
            PageReference pageRef = new PageReference(url);
            pageRef.setCookies(new Cookie[]{new Cookie('id', id, '/', 300, true), new Cookie('managerid', managerId,
'/', 300, true)});
            return pageRef;
        }

        public static PageReference createReference(String url){
            PageReference pageRef =  new PageReference(url);
            return pageRef;
        }

        public static String getId(){
            return ApexPages.currentPage().getCookies().get('id').getValue();
        }

        public static String getParam(String param){
            return ApexPages.currentPage().getCookies().get(param).getValue();
        }

    }

    public class PathConst {
```

```
    public static final String ALLUSERS = '/apex/ViewAllUsersForArea';
    public static final String ALLUSERSHISTORY = '/apex/AllUsersPaymentHistoryPage';
    public static final String BILLPDF = '/apex/BillPDF';
    public static final String CHANGERATES = '/apex/ChangeRates';
    public static final String CHANGEADDRESS = '/apex/ChangeAddress';
    public static final String CHANGEPAYMENT = '/apex/ChangeMonthlyPaymentPage';
    public static final String CONTACTMANAGER = '/apex/ContactManagerPage';
    public static final String LOGIN = '/apex/LoginPage';
    public static final String MANAGERWORKFLOW = '/apex/ManagerWorkflow';
    public static final String PAYMENTHISTORY = '/apex/PaymentHistoryPage';
    public static final String PAYMENT = '/apex/PaymentPage';
    public static final String REGISTER = '/apex/RegisterPage';
    public static final String UPDATEMETERS = '/apex/UpdateYourMeters';
    public static final String USERWORKFLOW = '/apex/UserWorkflow';
}

@RestResource(urlMapping='/Payment__c/*')
global class PaymentAPI{

    @HttpPost
    global static Id createPayment(String paidPrice, String userId, String isManager){
        Payment__c payment = new Payment__c();
        payment.Paid_Price__c = Decimal.valueOf(Double.valueOf(paidPrice));
        payment.User__c = userId;
        payment.byManager__c = Boolean.valueOf(isManager);
        insert payment;
        Customer__c currentUser = [SELECT Summary_debt_for_current_month__c,
                                   Gas_meters_reading__c, Electricity_meters_reading__c, Water_meters_reading__c
                                   FROM Customer__c
                                   WHERE id =:userId
                                   LIMIT 1];

        currentUser.Gas_meters_reading__c = 0.00;
        currentUser.Electricity_meters_reading__c = 0.00;
        currentUser.Water_meters_reading__c = 0.00;
        currentUser.Summary_debt_for_current_month__c -= payment.Paid_Price__c;
        System.debug(payment.Id+' - API ');
        update currentUser;

        return payment.Id;
    }
}

public class PaymentController {
    public Customer__c user{get;set;}
    public String paidPrice{get;set;}

    public PaymentController() {
        String userId = PageReferenceCreator.getId();
        user = [SELECT id, Name, Last_Name__c, Summary_debt__c, Email__c,
                (SELECT Id, Remain_debt__c FROM Payments__r
                ORDER BY CreatedDate DESC
                LIMIT 1)
            FROM Customer__c
            WHERE id = :userId
            LIMIT 1];
        paidPrice = '';


    }

    //payment process
    public PageReference createPayment(){
```

```
            if(paidPrice != ''){
                Map<String, String> params= new Map<String, String>();
                params.put('paidPrice', paidPrice);
                params.put('userId', String.valueOF(user.Id));
                params.put('isManager', 'false');
                HttpResponse response = new HttpResponse();

                //send request to create new payment

                try {
                    HttpResponse accessTokenResponse = PaymentControllerHelper.getAccessKey();
                    Map<String,Object> result = (Map<String, Object>)
JSON.deserializeUntyped(accessTokenResponse.getbody());
                    String accessToken =  String.valueOf(result.get('access_token'));
                    response = PaymentControllerHelper.createPayment(params, accessToken);
                }
                catch (Exception ex) {
                    MessageThrower.addMessageError(WarningConst.PAYMENTFAILED+'we here');
                    return null;
                }

                 //catching ok status
                if (response.getStatusCode() == 200) {
                    //reset meters


                    //sending bill
                    String paymentId =  (String)JSON.deserializeUntyped(response.getbody());
                    PaymentControllerHelper.sendBill(PaymentControllerHelper.createSuccessBill(user, paymentId));
                    //reference to user workflow
                    return PageReferenceCreator.createReference(PathConst.USERWORKFLOW, user.id);

                } else {
                    MessageThrower.addMessageError(WarningConst.PAYMENTFAILED+response.getStatusCode());
                    PaymentControllerHelper.sendBill(PaymentControllerHelper.createFailedBill(user));
                    return null;
                }

            }
            else {
                MessageThrower.addMessageWarning(WarningConst.EMPTYPRICE);
                return null;
            }
        }

        //cancelation
        public PageReference cancel(){
            return PageReferenceCreator.createReference(PathConst.USERWORKFLOW, user.Id);
        }

    }

    public class PaymentControllerHelper{

        //getting access token
        public static HttpResponse getAccessKey(){
            Http http = new Http();
            HttpRequest request = new HttpRequest();

            request.setEndpoint(APIParams.LOGINENDPOINT + '?' +
                        'client_id=' + APIParams.CLIENTID + '&'+
                        'grant_type=' + APIParams.GRANTTYPE + '&' +
                        'client_secret=' + APIParams.CLIENTSECRET + '&' +
```

```
                    'username=' + APIParams.USERNAME + '&' +
                    'password=' + APIParams.PASSWORD + APIParams.USERSECRET);

        request.setMethod('POST');
        request.setHeader('Content-Type', 'application/json');

        HttpResponse response = http.send(request);
        System.debug('return access');
        return response;

    }

    public static HttpResponse createPayment(Map<String, String> body, String token){
        Http http = new Http();
        HttpRequest request = new HttpRequest();

        request.setEndpoint(APIParams.PAYMENTAPIENDPOINT);
        request.setMethod('POST');
        request.setHeader('Authorization', 'Bearer '+token);
        request.setHeader('Content-Type', 'application/json;charset=UTF-8');
        request.setBody(JSON.serialize(body));

        HttpResponse response = http.send(request);
        System.debug('return status if created');
        return response;

    }

    //creating and sending bill
    public static void sendBill(Messaging.SingleEmailMessage bill){
          Messaging.sendEmail(new Messaging.SingleEmailMessage[] {bill});

    }

    public static Messaging.SingleEmailMessage createSuccessBill(Customer__c user, String paymentId){
        Messaging.SingleEmailMessage bill = new Messaging.SingleEmailMessage();
        System.debug(paymentId + ' - createMessage method');
        bill.setSubject('Payment details');
        bill.setToAddresses(new List<String>{user.Email__c});
        bill.setPlainTextBody('Hello, ' + user.Name + ' ' + user.Last_Name__c +
                        ', your payment was successful. In attachment you will find yor bill.');
        bill.setFileAttachments(new Messaging.EmailFileAttachment[] {createPDF(paymentId)});

        return bill;
    }

    public static Messaging.SingleEmailMessage createFailedBill(Customer__c user){
        Messaging.SingleEmailMessage bill = new Messaging.SingleEmailMessage();
        bill.setSubject('Payment details');
        bill.setToAddresses(new List<String>{user.Email__c});
        bill.setPlainTextBody('Hello, ' + user.Name + ' ' + user.Last_Name__c + 'your was not successful. Try
again');

        return bill;
    }

    public static Messaging.EmailFileAttachment createPDF(String Id){
        PageReference pdf = PageReferenceCreator.createReference(PathConst.BILLPDF);
        pdf.getParameters().put('id', id);
        System.debug(id + ' - create method');
        Blob page;

        if(Test.isRunningTest()){
```

```
            page = Blob.valueOf('test');
        }
        else{
            page = pdf.getContent();
        }

        // Create the email attachment
        Messaging.EmailFileAttachment attach = new Messaging.EmailFileAttachment();
        attach.setFileName('bill.pdf');
        attach.setBody(page);

        return attach;

    }

}

public class PaymentHistoryController {
    public List<Customer__c> users {get;set;}
    public List<Payment__c> payments{get;set;}
    public Customer__c user {get;set;}


    public PaymentHistoryController() {
        String userId = PageReferenceCreator.getId();
        users = [SELECT Name, Last_Name__c,
                    (SELECT Name, Paid_Price__c, CreatedDate, Remain_debt__c, byManager__c
                     FROM Payments__r
                     ORDER BY CreatedDate DESC)
                FROM Customer__c
                WHERE Id = :userId
                LIMIT 1];


        if (users.isEmpty() == false) {
            user = users.get(0);
        }

        payments = new List<Payment__c>();
        for (Payment__c payment : user.Payments__r) {
            payments.add(payment);
        }
    }



    public PageReference back(){
        return PageReferenceCreator.createReference(PathConst.USERWORKFLOW, user.Id);
    }
}

public class RegisterController {
    public Customer__c user{get; set;}
    public String repeatPasswordInput{get;set;}
    public String oblast{get;set;}



    public RegisterController() {
        user = new Customer__c();
        repeatPasswordInput = '';
        oblast = '';
    }
```

```
    public PageReference createNewUser(){

        if (RegisterHelper.newUserValidation(user) && isInputCorrect()) {

            RegisterHelper.createUser(user);
            return PageReferenceCreator.createReference(PathConst.USERWORKFLOW,  user.Id);

        }
        else if(!RegisterHelper.newUserValidation(user)){
            MessageThrower.addMessageWarning(WarningConst.USEREXIST);
            return null;
        }
        else {
            return null;
        }


    }

    public List<SelectOption> getAddresses(){
        return GetListOfAddresses.getAddresses();
    }

    public Boolean isInputCorrect(){
        Boolean status = false;

        Matcher nameSpace = Pattern.compile('^[a-z|A-Z]+').matcher(user.Name);
        Matcher lastNameSpace = Pattern.compile('^[a-z|A-Z]+').matcher(user.Last_Name__c);
        Matcher password = Pattern.compile('^[a-z|A-Z|0-9]+').matcher(user.Password__c);

        if (nameSpace.matches() &&
            lastNameSpace.matches() &&
            password.matches() &&
            user.Password__c == repeatPasswordInput) {
            status = true;
        }
        else if(user.Password__c.toLowerCase() != repeatPasswordInput.toLowerCase()){
            MessageThrower.addMessageError(WarningConst.PASSWORDINPUTERROR);
        }
        else {
            MessageThrower.addMessageError(WarningConst.INPUTERROR);
        }
        return status;

    }


}

public class RegisterHelper{

    public static Boolean newUserValidation(Customer__c user) {
        Boolean canCreate = false;
        List<Customer__c> existingUser = [SELECT Id, Email__c, Password__c
                            FROM Customer__c
                            WHERE Email__c = :user.Email__c
                            LIMIT 1];
        if (existingUser.isEmpty()) {
            canCreate = true;
            return canCreate;
        }
        else{
```

```
            return canCreate;
        }
    }

    public static void createUser(Customer__c user){
        insert user;
    }
}

global class SetNewMetersSchedule implements Schedulable{

    global void execute(SchedulableContext sc){
        //get all users
        List<Customer__c> users = [SELECT Name, Last_Name__c, Email__c, LastModifiedDate,
Gas_meters_reading__c, Electricity_meters_reading__c, Water_meters_reading__c
                                   FROM Customer__c];

        //creating list with days when user can pay
        List<Integer> valideDates = new List<Integer>();
        for (Integer i = 1; i < 10; i++) {
            valideDates.add(i);
        }
        //Check when users updated they meters
        for (Customer__c user : users) {
            if (!valideDates.contains(user.LastModifiedDate.day())) {

                Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
                email.toAddresses = new String[]{user.Email__c};
                email.setSubject('Warning. Input your meters');
                email.optOutPolicy = 'FILTER';
                email.plainTextBody = user.Name+' '+ user.Last_Name__c + ', update your meters or standart meters
would set in 10th day of current month\n'+
                'Best regards,\n'+
                'Evil corporation';

                Messaging.sendEmail(new List<Messaging.SingleEmailMessage> {email});
            }
            else if (Date.today().day() == 10) {
                user.Gas_meters_reading__c *= 2;
                user.Electricity_meters_reading__c *= 2;
                user.Water_meters_reading__c *= 2;

            }
        }
        update users;
    }
}

@isTest
public class TestAllUsersPaymentHistoryController{
    @isTest
    static void testAllUsersPaymentHistoryInit(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Region__c region = (Region__c)testData.get('region');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.ALLUSERSHISTORY, region.id));
        AllUsersPaymentHistoryController allUsersPaymentHistoryInstance = new
AllUsersPaymentHistoryController();
        //check user
        System.assertEquals(user.Id, allUsersPaymentHistoryInstance.regionUsers[0].Id);

    }
```

```
        @isTest
        static void testGetUsers(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Customer__c user = (Customer__c)testData.get('user');
            Region__c region = (Region__c)testData.get('region');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.ALLUSERSHISTORY, region.id));
            AllUsersPaymentHistoryController allUsersPaymentHistoryInstance = new
AllUsersPaymentHistoryController();

            List<SelectOption> users = allUsersPaymentHistoryInstance.getUsers();

            System.assertEquals(user.Id, users[0].getValue());
        }

        @isTest
        static void testRerender(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Payment__c payment = (Payment__c)testData.get('payment');
            Customer__c user = (Customer__c)testData.get('user');
            Region__c region = (Region__c)testData.get('region');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.ALLUSERSHISTORY, region.id));
            AllUsersPaymentHistoryController allUsersPaymentHistoryInstance = new
AllUsersPaymentHistoryController();
            allUsersPaymentHistoryInstance.userId = (String)user.Id;

            PageReference expectedPage = allUsersPaymentHistoryInstance.rerender();
            //check page
            System.assertEquals(null, expectedPage);

            //check selected user
            System.assertEquals(user.Id, allUsersPaymentHistoryInstance.selectedUser.Id, 'user invalid');

            //check payment
            System.assertEquals(payment.Id, allUsersPaymentHistoryInstance.selectedUser.payments__r[0].Id,
'payment invalid');


        }

        @isTest
        static void testBackReference(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Customer__c manager = (Customer__c)testData.get('manager');
            Region__c region = (Region__c)testData.get('region');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.ALLUSERSHISTORY, region.id,
manager.Id));
            AllUsersPaymentHistoryController allUsersPaymentHistoryInstance = new
AllUsersPaymentHistoryController();

            PageReference expectedPage = allUsersPaymentHistoryInstance.back();

            //check url
            System.assertEquals('/apex/ManagerWorkflow', expectedPage.getUrl());

        }
    }

    @isTest
    public class TestBillController{
        @isTest
        static void testBillInit(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Payment__c payment = (Payment__c)testData.get('payment');
```

```
        PageReference currentPage = new PageReference(PathConst.BILLPDF);
        currentPage.getParameters().put('id', payment.Id);
        Test.setCurrentPage(currentPage);

        BillController billInstance = new BillController();

        //check bill
        System.assertEquals(payment.id, billInstance.bill.Id);

    }
}

@isTest
public class TestChangeAddressController{
    @isTest
    static void testChangeAddressInit(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CHANGEADDRESS, user.id));
        ChangeAddressController changeAddressInstance = new ChangeAddressController();
        //check user
        System.assertEquals(user.Id, changeAddressInstance.user.Id);

    }

    @isTest
    static void testGetRegions(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CHANGEADDRESS, user.id));
        ChangeAddressController changeAddressInstance = new ChangeAddressController();
        List<SelectOption> options = changeAddressInstance.getAddresses();
        Address__c address = [SELECT Id, Name
                              FROM Address__c
                              LIMIT 1];

        //get regions
        System.assertEquals(address.Id, options[0].getValue());
    }

    @isTest
    static void testSaveReference(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CHANGEADDRESS, user.id));
        ChangeAddressController changeAddressInstance = new ChangeAddressController();

        PageReference expectedPage = changeAddressInstance.save();
        //check url
        System.assertEquals('/apex/UserWorkflow', expectedPage.getUrl());

        //check user
        System.assertEquals(user.Id, expectedPage.getCookies().get('id').getValue());

    }
}

@isTest
public class TestChangeMonthlyPaymentController{
    @isTest
    static void testChangeMonthlyPaymentInit(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Region__c region = (Region__c)testData.get('region');
```

```
            Customer__c manager = (Customer__c)testData.get('manager');
            Customer__c user = (Customer__c)testData.get('user');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CHANGEPAYMENT, region.id,
manager.id));

            ChangeMonthlyPaymentController changeMonthlyPaymentInstance = new
ChangeMonthlyPaymentController();

            //check region
            System.assertEquals(user.Name, changeMonthlyPaymentInstance.regionUsers[0].Name);
        }

        @isTest
        static void testGetUsers(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Region__c region = (Region__c)testData.get('region');
            Customer__c user = (Customer__c)testData.get('user');
            Customer__c manager = (Customer__c)testData.get('manager');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CHANGEPAYMENT, region.id,
manager.id));

            ChangeMonthlyPaymentController changeMonthlyPaymentInstance = new
ChangeMonthlyPaymentController();
            List<SelectOption> users = changeMonthlyPaymentInstance.getUsers();

            //check user
            System.assertEquals(user.Name + ' ' + user.Last_Name__c, users[0].getLabel());
        }

        @isTest
        static void testRerender(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Region__c region = (Region__c)testData.get('region');
            Customer__c user = (Customer__c)testData.get('user');
            Customer__c manager = (Customer__c)testData.get('manager');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CHANGEPAYMENT, region.id,
manager.id));

            ChangeMonthlyPaymentController changeMonthlyPaymentInstance = new
ChangeMonthlyPaymentController();
            changeMonthlyPaymentInstance.userId = user.Id;
            PageReference expectedPage = changeMonthlyPaymentInstance.rerender();

            //check page
            System.assertEquals(null, expectedPage);
            //check user
            System.assertEquals(user.Id, changeMonthlyPaymentInstance.selectedUser.Id, 'userInvalid');
        }

        @isTest
        static void testBackReference(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Region__c region = (Region__c)testData.get('region');
            Customer__c manager = (Customer__c)testData.get('manager');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CHANGEPAYMENT, region.id,
manager.id));

            ChangeMonthlyPaymentController changeMonthlyPaymentInstance = new
ChangeMonthlyPaymentController();
            PageReference expectedPage = changeMonthlyPaymentInstance.back();

            //check url
            System.assertEquals('/apex/ManagerWorkflow', expectedPage.getUrl());
```

```
            //check user
            System.assertEquals(manager.Id, expectedPage.getCookies().get('id').getValue());
        }

        @isTest
        static void testSave(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Region__c region = (Region__c)testData.get('region');
            Customer__c user = (Customer__c)testData.get('user');
            Customer__c manager = (Customer__c)testData.get('manager');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CHANGEPAYMENT, region.id,
manager.id));

            ChangeMonthlyPaymentController changeMonthlyPaymentInstance = new
ChangeMonthlyPaymentController();
            changeMonthlyPaymentInstance.selectedUser = user;
            changeMonthlyPaymentInstance.paidPrice = '6';

            Test.startTest();
            PageReference expectedPage = changeMonthlyPaymentInstance.save();
            Integer dmlInvocations = Limits.getDmlStatements();
            Test.stopTest();

            //check page
            System.assertEquals(null, expectedPage);
            //check user
            System.assertEquals(user.Id, changeMonthlyPaymentInstance.selectedUser.Id, 'userInvalid');
            //check new summary debt
            System.assertEquals(7.00,
changeMonthlyPaymentInstance.selectedUser.Summary_debt_for_current_month__c);
        }

    }

    @isTest
    public class TestChangeRatesController{
        @isTest
        static void testChangeRatesInit(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Region__c region = (Region__c)testData.get('region');
            Customer__c manager = (Customer__c)testData.get('manager');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CHANGERATES, region.id,
manager.Id));

            ChangeRatesController changeRatesInstance = new ChangeRatesController();

            //check region
            System.assertEquals(region.Id, changeRatesInstance.region.Id);

        }

        @isTest
        static void testSaveReference(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Customer__c manager = (Customer__c)testData.get('manager');
            Region__c region = (Region__c)testData.get('region');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CHANGERATES, region.id,
manager.Id));

            ChangeRatesController changeRatesInstance = new ChangeRatesController();

            changeRatesInstance.region.Gas_rate__c = 5.00;
            changeRatesInstance.region.Electricity_rate__c = 5.00;
```

```
            changeRatesInstance.region.Water_rate__c = 5.00;

            Test.startTest();
            PageReference expectedPage = changeRatesInstance.save();
            Integer dmlInvocations = Limits.getDmlStatements();
            Test.stopTest();

            //check updating
            System.assertEquals(1, dmlInvocations, 'Record wasn\'t updated');
            //check url
            System.assertEquals(null, expectedPage);

        }

        @isTest
        static void testBackReference(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Customer__c manager = (Customer__c)testData.get('manager');
            Region__c region = (Region__c)testData.get('region');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CHANGERATES, region.id,
manager.Id));

            ChangeRatesController changeRatesInstance = new ChangeRatesController();

            PageReference expectedPage = changeRatesInstance.back();
            //check url
            System.assertEquals('/apex/ManagerWorkflow', expectedPage.getUrl());
            //check user
            System.assertEquals(manager.Id, expectedPage.getCookies().get('id').getValue());

        }
    }

    @isTest
    public class TestContactManagerController{
        @isTest
        static void testContactManagerInit(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Customer__c user = (Customer__c)testData.get('user');
            Customer__c manager = (Customer__c)testData.get('manager');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CONTACTMANAGER, user.id));

            ContactManagerController contactManagerInstance = new ContactManagerController();

            //check user
            System.assertEquals(user.Id, contactManagerInstance.user.Id, 'user invalid');
            //check manager
            System.assertEquals(manager.Id, contactManagerInstance.manager.Id, 'manager invalid');
        }

        @isTest
        static void testSendToManagerSuccess(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Customer__c user = (Customer__c)testData.get('user');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CONTACTMANAGER, user.id));

            ContactManagerController contactManagerInstance = new ContactManagerController();

            contactManagerInstance.emailText = 'test text';

            Test.startTest();
            PageReference expectedPage = contactManagerInstance.sendToManager();
            Integer invocations = Limits.getEmailInvocations();
```

```
        Test.stopTest();

        //check sending message
        System.assertEquals(1, invocations);
        //check url
        System.assertEquals('/apex/UserWorkflow', expectedPage.getUrl());
        //check user
        System.assertEquals(user.Id, expectedPage.getCookies().get('id').getValue());

    }

    @isTest
    static void testSendToManagerFailed(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CONTACTMANAGER, user.id));

        ContactManagerController contactManagerInstance = new ContactManagerController();

        Test.startTest();
        PageReference expectedPage = contactManagerInstance.sendToManager();
        Integer invocations = Limits.getEmailInvocations();
        Test.stopTest();

        //check sending message
        System.assertEquals(0, invocations);
        //check url
        System.assertEquals(null, expectedPage);
    }

    @isTest
    static void testCancelReference(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.CONTACTMANAGER, user.id));

        ContactManagerController contactManagerInstance = new ContactManagerController();

        PageReference expectedPage = contactManagerInstance.cancel();

        //check url
        System.assertEquals('/apex/UserWorkflow', expectedPage.getUrl());
        //check user
        System.assertEquals(user.Id, expectedPage.getCookies().get('id').getValue());


    }
}

@isTest
public class TestDataSet{

    public static Map<String, SObject> createTestData(){

        Map<String, SObject> testData = new Map<String, SObject>();

        //test region

        Region__c region = new Region__c();
        region.Name = 'testRegion';
        region.Gas_rate__c = 2.00;
        region.Electricity_rate__c = 2.00;
        region.Water_rate__c = 2.00;
```

```
insert region;

//test address

Address__c address = new Address__c();
address.Region__c = region.Id;
address.Name = 'Test Address';
insert address;

//test manager

Customer__c manager = new Customer__c();
manager.Name = 'Testy';
manager.Last_Name__c = 'testy';
manager.Email__c = 'oleksii.llin@sparkybit.com';
manager.Password__c = '1234';
manager.Address__c = address.Id;
manager.isManager__c = true;
insert manager;

//test user

Customer__c user = new Customer__c();
user.Name = 'Test';
user.Last_Name__c = 'test';
user.Email__c = 'master.iljin97@gmail.com';
user.Password__c = '1234';
user.Gas_meters_reading__c = 3.00;
user.Electricity_meters_reading__c = 3.00;
user.Water_meters_reading__c = 3.00;
user.Summary_debt_for_current_month__c = region.Gas_rate__c * user.Gas_meters_reading__c +
                                region.Electricity_rate__c * user.Electricity_meters_reading__c +
                                region.Water_rate__c + user.Water_meters_reading__c;
user.Address__c = address.Id;
insert user;

//test payment

Payment__c payment = new Payment__c();
payment.User__c = user.Id;
payment.Paid_Price__c = 10.00;
insert payment;

//add manager to region
user.Summary_debt_for_current_month__c -= payment.Paid_Price__c;

//set all data to map

testData.put('region', region);
testData.put('manager', manager);
testData.put('user', user);
testData.put('payment', payment);
testData.put('address', address);

//updating all data

List<SObject> objects = new List<SObject>();
objects.addAll(testData.values());
update objects;

return testData;
```

```
    }
}

@isTest
public class TestLimitDebtTrigger{
    @IsTest
    static void testSendingEmail(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        user.Summary_debt_for_current_month__c = 7600.00;

        Test.startTest();
        Database.SaveResult result =  Database.update(user);
        Integer invocations = Limits.getEmailInvocations();
        Test.stopTest();

        //check email sends
        System.assertEquals(1, invocations);

    }

    @IsTest
    static void testNotSendinEmail(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        user.Summary_debt_for_current_month__c = 1500.00;

        Test.startTest();
        Database.SaveResult results =  Database.update(user);
        Integer invocations = Limits.getEmailInvocations();
        Test.stopTest();

        //check email sends
        System.assertEquals(0, invocations);
    }
}

@isTest
public class TestLoginController{
    @isTest
    static void testLoginAsUserSuccess(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');

        LoginController loginInstance = new LoginController();
        loginInstance.login = user.Email__c;
        loginInstance.password = user.PassWord__c;

        PageReference expectedPage = loginInstance.getCustomer();
        String userId = expectedPage.getCookies().get('id').getValue();

        //check path
        System.assertEquals('/apex/UserWorkflow', expectedPage.getUrl());
        //user found
        System.assertEquals(user.Id, userId);

    }
    @isTest
    static void testLoginAsUserFailed(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');

        LoginController loginInstance = new LoginController();
```

```
            PageReference expectedPage = loginInstance.getCustomer();

            //check page
            System.assertEquals(null, expectedPage);

        }
        @isTest
        static void testRegisterReference(){
            LoginController loginInstance = new LoginController();
            PageReference expectedPage = loginInstance.register();

            //check path
            System.assertEquals('/apex/RegisterPage', expectedPage.getUrl());

        }


        @isTest
        static void testLoginAsManagerSuccess(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Customer__c manager = (Customer__c)testData.get('manager');

            LoginController loginInstance = new LoginController();
            loginInstance.login = manager.Email__c;
            loginInstance.password = manager.PassWord__c;

            PageReference expectedPage = loginInstance.managerWorkflow();
            String managerId = expectedPage.getCookies().get('id').getValue();

            //check path
            System.assertEquals('/apex/ManagerWorkflow', expectedPage.getUrl());
            //user found
            System.assertEquals(manager.Id, managerId);

        }
        @isTest
        static void testLoginAsManagerFailed(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Customer__c manager = (Customer__c)testData.get('manager');

            LoginController loginInstance = new LoginController();
            PageReference expectedPage = loginInstance.managerWorkflow();

            //check url
            System.assertEquals(null, expectedPage);
        }

    }

    @isTest
    public class TestManagerWorkflowController {
        @isTest
        static void testManagerWorkflowInit(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Customer__c manager = (Customer__c)testData.get('manager');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.MANAGERWORKFLOW,
manager.id));

            ManagerWorkflowController managerWorkflowInstance = new ManagerWorkflowController();

            //check manager
            System.assertEquals(manager.Id, managerWorkflowInstance.manager.Id);
        }
```

```
    @isTest
    static void testChangeRatesReference(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c manager = (Customer__c)testData.get('manager');
        Region__c region = (Region__c)testData.get('region');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.MANAGERWORKFLOW,
manager.id));

        ManagerWorkflowController managerWorkflowInstance = new ManagerWorkflowController();
        PageReference expectedPage = managerWorkflowInstance.changeRates();

        //check url
        System.assertEquals('/apex/ChangeRates', expectedPage.getUrl());
        //check user
        System.assertEquals(region.Id, expectedPage.getCookies().get('id').getValue());
    }

    @isTest
    static void testViewAllUsersReference(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c manager = (Customer__c)testData.get('manager');
        Region__c region = (Region__c)testData.get('region');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.MANAGERWORKFLOW,
manager.id));

        ManagerWorkflowController managerWorkflowInstance = new ManagerWorkflowController();
        PageReference expectedPage = managerWorkflowInstance.viewAllUsers();

        //check url
        System.assertEquals('/apex/ViewAllUsersForArea', expectedPage.getUrl());
        //check user
        System.assertEquals(region.Id, expectedPage.getCookies().get('id').getValue());
    }

    @isTest
    static void testUsersPaymentHistoryReference(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c manager = (Customer__c)testData.get('manager');
        Region__c region = (Region__c)testData.get('region');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.MANAGERWORKFLOW,
manager.id));

        ManagerWorkflowController managerWorkflowInstance = new ManagerWorkflowController();
        PageReference expectedPage = managerWorkflowInstance.usersPaymentHistory();

        //check url
        System.assertEquals('/apex/AllUsersPaymentHistoryPage', expectedPage.getUrl());
        //check user
        System.assertEquals(region.Id, expectedPage.getCookies().get('id').getValue());
    }

    @isTest
    static void testUpdateMonthlyPaymentReference(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c manager = (Customer__c)testData.get('manager');
        Region__c region = (Region__c)testData.get('region');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.MANAGERWORKFLOW,
manager.id));

        ManagerWorkflowController managerWorkflowInstance = new ManagerWorkflowController();
        PageReference expectedPage = managerWorkflowInstance.updateMonthlyPayment();
```

```
        //check url
        System.assertEquals('/apex/ChangeMonthlyPaymentPage', expectedPage.getUrl());
        //check user
        System.assertEquals(region.Id, expectedPage.getCookies().get('id').getValue());
    }
}

@isTest
public class TestPaymentAPI{
    @isTest
    static void testCreatePayment(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        String paidPrice = '2';
        String isManager = 'false';

        Id newPaymentId = PaymentAPI.createPayment(paidPrice, user.Id, isManager);

        Payment__c newPayment = [SELECT Paid_Price__c, User__c
                                 FROM Payment__c
                                 WHERE Id = :newPaymentId];

        //check price
        System.assertEquals(2.00, newPayment.Paid_Price__c, 'invalid Paid Price');
        //check user
        System.assertEquals(user.Id, newPayment.User__c, 'invalid user');

    }
}

@isTest
public class TestPaymentController{

    @isTest
    static void testCancelReference(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.PAYMENT, user.id));

        PaymentController paymentInstance = new PaymentController();
        PageReference expectedPage = paymentInstance.cancel();

        //check url
        System.assertEquals('/apex/UserWorkflow', expectedPage.getUrl());
        //check current user
        System.assertEquals(user.id, expectedPage.getCookies().get('id').getValue());

    }

    @isTest
    static void testCreatePaymentWithEmptyPaidPrice(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.PAYMENT, user.id));

        PaymentController paymentInstance = new PaymentController();
        PageReference expectedPage = paymentInstance.createPayment();

        //check page
        System.assertEquals(null, expectedPage);

    }
```

```
    @isTest
    static void testCreatePaymentWithError(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.PAYMENT, user.id));

        PaymentController paymentInstance = new PaymentController();
        paymentInstance.paidPrice = '10';
        PageReference expectedPage = paymentInstance.createPayment();

        //check page
        System.assertEquals(null, expectedPage);

    }
}

@isTest
public class TestPaymentControllerHelper{

    public static final String EXPECTING_TOKEN = '1234567890QWERTY';

    @isTest
    static void testGetAccessToken(){
        Test.setMock(HttpCalloutMock.class, new TestPaymentControllerMock());
        HttpResponse response = PaymentControllerHelper.getAccessKey();

        Map<String,Object> result = (Map<String, Object>) JSON.deserializeUntyped(response.getbody());
        String accessToken =  String.valueOf(result.get('access_token'));

        //check token
        System.assertEquals(EXPECTING_TOKEN, accessToken);
    }

    @isTest
    static void testCreatePayment(){
        Test.setMock(HttpCalloutMock.class, new TestPaymentControllerMock());
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Map<String, String> body = new Map<String, String>();
        body.put('paidPrice', '10');
        body.put('userId', user.Id);

        Test.startTest();
        HttpResponse response = PaymentControllerHelper.createPayment(body, EXPECTING_TOKEN);
        Test.stopTest();

        //check status
        System.assertEquals(200, response.getStatusCode());
    }

    @isTest
    static void testSendBillWithSuccessStatus(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Payment__c payment = (Payment__c)testData.get('payment');

        Test.startTest();
        PaymentControllerHelper.sendBill(PaymentControllerHelper.createSuccessBill(user, payment.Id));
        Integer emailInvocation = Limits.getEmailInvocations();
        Test.stopTest();

        //check email sends
        System.assertEquals(1, emailInvocation);
```

```
    }

    @isTest
    static void testSendBillWithFailedStatus(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');

        Test.startTest();
        PaymentControllerHelper.sendBill(PaymentControllerHelper.createFailedBill(user));
        Integer emailInvocation = Limits.getEmailInvocations();
        Test.stopTest();

        //check email sends
        System.assertEquals(1, emailInvocation);

    }

}

@isTest
global class TestPaymentControllerMock implements HttpCalloutMock {
    global HttpResponse respond(HttpRequest request){
        String checkTokenEndpoint = 'https://login.salesforce.com/services/oauth2/token?'+

'client_id=3MVG91BJr_0ZDQ4tV15NHF357DzlGSBl3Gu0fcthIPZbkjvXnb24sV7VIUGzdbzWgfalcMpxwXR.PdqLwvXHg&'+

'grant_type=password&client_secret=7D71224CE7E2FC164BBD95A13B07C672A1A16A17EB28E2F30146EEAD7B3ABC56&'+
                                'username=oleksii.ilin@alexinc.com&'+
                                'password=Apple2016B3aM9Cn9knAzPNXAR7UG6foq';

        String checkPaymentEndpoint = 'https://eu29.salesforce.com/services/apexrest/Payment__c/';

        Map<String, String> fakeResponseBody = new Map<String, String>();

        HttpResponse response = new HttpResponse();

        response.setHeader('Content-Type', 'application/json');
        response.setStatusCode(200);

        if (request.getEndpoint() == checkTokenEndpoint) {
            fakeResponseBody.put('access_token', '1234567890QWERTY');
            response.setBody(JSON.serialize(fakeResponseBody));
        }
        else if(request.getEndpoint() == checkPaymentEndpoint){
            fakeResponseBody.put('id', 'fake id');
        }

        return response;
    }
}

@isTest
public class TestPaymentHistoryController{
    @isTest
    static void testListCreation(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Payment__c payment = (Payment__c)testData.get('payment');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.PAYMENTHISTORY, user.id));
```

```
        PaymentHistoryController paymentHistoryInstance = new PaymentHistoryController();
        Payment__c actualPayment = paymentHistoryInstance.payments[0];

        //check payment
        System.assertEquals(payment.Id, actualPayment.Id);
        //check user
        System.assertEquals(user.Id, paymentHistoryInstance.user.Id);

    }

    @isTest
    static void testBackReference(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Payment__c payment = (Payment__c)testData.get('payment');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.PAYMENTHISTORY, user.id));

        PaymentHistoryController paymentHistoryInstance = new PaymentHistoryController();
        PageReference expectedPage = paymentHistoryInstance.back();

        //check url
        System.assertEquals('/apex/UserWorkflow' , expectedPage.getUrl());
        //check current user
        System.assertEquals(user.id, expectedPage.getCookies().get('id').getValue());

    }
}

@isTest
public class TestRegisterController {

    @isTest
    static void testCreateNewUser(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Address__c address = (Address__c)testData.get('address');

        RegisterController registerInstance = new RegisterController();

        //filling user fields
        registerInstance.user.Name = 'Test';
        registerInstance.user.Last_Name__c = 'test';
        registerInstance.user.Email__c = 'any@gmail.com';
        registerInstance.user.Password__c = '1234';
        registerInstance.user.Gas_meters_reading__c = 3.00;
        registerInstance.user.Electricity_meters_reading__c = 3.00;
        registerInstance.user.Water_meters_reading__c = 3.00;
        registerInstance.user.Address__c = address.id;
        registerInstance.repeatPasswordInput = '1234';

        PageReference expectedPage = registerInstance.createNewUser();
        String userId = expectedPage.getCookies().get('id').getValue();

        Customer__c createdUser = [SELECT Id, Name
                                   FROM Customer__c
                                   WHERE Id = :userId
                                   LIMIT 1];

        //check path
        System.assertEquals('/apex/UserWorkflow', expectedPage.getUrl());
        //user found
        System.assertEquals(createdUser.Id, userId);

    }
```

```
@isTest
static void testCreateNewUserFailed(){
    Map<String, SObject> testData = TestDataSet.createTestData();
    Address__c address = (Address__c)testData.get('address');

    RegisterController registerInstance = new RegisterController();

    registerInstance.user.Name = 'Te!@$st';
    registerInstance.user.Last_Name__c = 'test';
    registerInstance.user.Email__c = 'any@gmail.com';
    registerInstance.user.Password__c = '1234';
    registerInstance.user.Gas_meters_reading__c = 3.00;
    registerInstance.user.Electricity_meters_reading__c = 3.00;
    registerInstance.user.Water_meters_reading__c = 3.00;
    registerInstance.user.Address__c = address.id;
    registerInstance.repeatPasswordInput = '1234';

    PageReference expectedPage = registerInstance.createNewUser();

    //check page
    System.assertEquals(null, expectedPage);

}

@isTest
static void testCreateNewUserPasswordMissmatch(){
    RegisterController registerInstance = new RegisterController();

    registerInstance.user.Name = 'Test';
    registerInstance.user.Last_Name__c = 'test';
    registerInstance.user.Email__c = 'any@gmail.com';
    registerInstance.user.Password__c = '1234';
    registerInstance.user.Gas_meters_reading__c = 3.00;
    registerInstance.user.Electricity_meters_reading__c = 3.00;
    registerInstance.user.Water_meters_reading__c = 3.00;

    //repeatpassword mistaken enterance
    registerInstance.repeatPasswordInput = '12345';

    PageReference expectedPage = registerInstance.createNewUser();

    //check page
    System.assertEquals(null, expectedPage);

}

@isTest
static void testCreateNewExistingUser(){
    Map<String, SObject> testData = TestDataSet.createTestData();

    RegisterController registerInstance = new RegisterController();

    registerInstance.user.Name = 'Test';
    registerInstance.user.Last_Name__c = 'test';
    registerInstance.user.Email__c = 'master.iljin97@gmail.com';
    registerInstance.user.Password__c = '1234';
    registerInstance.user.Gas_meters_reading__c = 3.00;
    registerInstance.user.Electricity_meters_reading__c = 3.00;
    registerInstance.user.Water_meters_reading__c = 3.00;
    registerInstance.repeatPasswordInput = '1234';

    PageReference expectedPage = registerInstance.createNewUser();
```

```
        //check page
        System.assertEquals(null, expectedPage);

    }

    @isTest
    static void testGetAddresses(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Address__c address = (Address__c)testData.get('address');

        RegisterController registerInstance = new RegisterController();
        List<SelectOption> options = registerInstance.getAddresses();

        //get address
        System.assertEquals(address.Id, options[0].getValue());
    }


}

@isTest
public class TestSetNewMetersSchedule{

    @isTest
    static void testMessageNotSend(){
        String CRON_EXP = '0 0 0 6-10 * ? 2020';
        Map<String, SObject> testData = TestDataSet.createTestData();

        Test.startTest();
        SetNewMetersSchedule scheduledJob = new SetNewMetersSchedule();
        String jobId = System.schedule('TestSetNewMetersSchedule', CRON_EXP, scheduledJob);
        Integer Invocations = Limits.getEmailInvocations();
        Test.stopTest();

        CronTrigger cronTrigger = [SELECT Id, CronExpression, TimesTriggered, NextFireTime
                                   FROM CronTrigger
                                   WHERE id = :jobId];

        // Verify the expressions are the same
        System.assertEquals(CRON_EXP, cronTrigger.CronExpression);
        // Verify the job has not run
        System.assertEquals(0, cronTrigger.TimesTriggered);
        // Verify the next time the job will run
        System.assertEquals('2020-03-06 00:00:00', String.valueOf(cronTrigger.NextFireTime));
        // Verify the scheduled job hasn't run yet.
        System.assertEquals(0 , Invocations, 'sent email');

    }

}

@isTest
public class TestUpdateMetersController{
    @isTest
    static void testUpdateMetersInit(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.UPDATEMETERS, user.id));

        UpdateMetersController updateMetersInstance = new UpdateMetersController();

        //check user
```

```
            System.assertEquals(user.Id, updateMetersInstance.user.Id);

        }

        @isTest
        static void testSaveReference(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Customer__c user = (Customer__c)testData.get('user');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.UPDATEMETERS, user.id));

            UpdateMetersController updateMetersInstance = new UpdateMetersController();
            PageReference expectedPage = updateMetersInstance.save();

            //check url
            System.assertEquals('/apex/UserWorkflow', expectedPage.getUrl());
            //check user
            System.assertEquals(user.Id, expectedPage.getCookies().get('id').getValue());

        }
    }

    @isTest
    public class TestUserWorkflowController{
        @isTest
        static void testIsDebtCheck(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Customer__c user = (Customer__c)testData.get('user');
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.USERWORKFLOW, user.id));

            UserWorkflowController workflowInstance = new UserWorkflowController();
            PageReference expectedPage = workflowInstance.toChangeAddress();

            //Check debt
            System.assertEquals(user.Summary_debt_for_current_month__c,
workflowInstance.user.Summary_debt__c);
            //check current user
            System.assertEquals(null, expectedPage);

        }

        @isTest
        static void testIsDebtCheckWhenNull(){
            Map<String, SObject> testData = TestDataSet.createTestData();
            Customer__c user = (Customer__c)testData.get('user');
            user.Summary_debt_for_current_month__c = 0.00;
            update user;
            Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.USERWORKFLOW, user.id));

            UserWorkflowController workflowInstance = new UserWorkflowController();
            PageReference expectedPage = workflowInstance.toChangeAddress();

            //check debt
            System.assertEquals(0.00, workflowInstance.user.Summary_debt__c);
            //check url
            System.assertEquals('/apex/ChangeAddress', expectedPage.getUrl());
            //check current user
            System.assertEquals(user.id, expectedPage.getCookies().get('id').getValue());

        }

        @isTest
        static void testToUpdateMetersReference(){
            Map<String, SObject> testData = TestDataSet.createTestData();
```

```
        Customer__c user = (Customer__c)testData.get('user');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.USERWORKFLOW, user.id));

        UserWorkflowController workflowInstance = new UserWorkflowController();
        PageReference expectedPage = workflowInstance.toUpdateMeters();

        //check url
        System.assertEquals('/apex/UpdateYourMeters', expectedPage.getUrl());
        //check current user
        System.assertEquals(user.id, expectedPage.getCookies().get('id').getValue());

    }

    @isTest
    static void testPaymentReference(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.USERWORKFLOW, user.id));

        UserWorkflowController workflowInstance = new UserWorkflowController();
        PageReference expectedPage = workflowInstance.payment();

        //check url
        System.assertEquals('/apex/PaymentPage', expectedPage.getUrl());
        //check current user
        System.assertEquals(user.id, expectedPage.getCookies().get('id').getValue());

    }

    @isTest
    static void testContactManagerReference(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Customer__c manager = (Customer__c)testData.get('manager');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.USERWORKFLOW, user.id));

        UserWorkflowController workflowInstance = new UserWorkflowController();
        PageReference expectedPage = workflowInstance.contactManager();

        //check url
        System.assertEquals('/apex/ContactManagerPage', expectedPage.getUrl());
        //check current user
        System.assertEquals(user.id, expectedPage.getCookies().get('id').getValue());

    }

    @isTest
    static void testPaymentHistoryReference(){
        Map<String, SObject> testData = TestDataSet.createTestData();
        Customer__c user = (Customer__c)testData.get('user');
        Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.USERWORKFLOW, user.id));

        UserWorkflowController workflowInstance = new UserWorkflowController();
        PageReference expectedPage = workflowInstance.paymentHistory();

        //check url
        System.assertEquals('/apex/PaymentHistoryPage', expectedPage.getUrl());
        //check current user
        System.assertEquals(user.id, expectedPage.getCookies().get('id').getValue());

    }
}
```

```
@isTest
public class TestUserWorkflowHelper{
   @isTest
   static void testDebtCalculatorWithNull(){
      System.assertEquals(0.00, UserWorkflowHelper.debtCalculator(12, 0.00));

   }

}

@isTest
public class TestViewAllUsersForAreaController{
   @isTest
   static void testViewAllUsersForAreaInit(){
      Map<String, SObject> testData = TestDataSet.createTestData();
      Region__c region = (Region__c)testData.get('region');
      Customer__c user = (Customer__c)testData.get('user');
      Customer__c manager = (Customer__c)testData.get('manager');
      Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.ALLUSERSHISTORY, region.Id,
manager.id));

      ViewAllUsersForAreaController viewAllUsersForAreaInstance = new ViewAllUsersForAreaController();

      //check user
      System.assertEquals(user.Id, viewAllUsersForAreaInstance.regionUsers[0].Id);

   }

   @isTest
   static void testGetUsers(){
      Map<String, SObject> testData = TestDataSet.createTestData();
      Region__c region = (Region__c)testData.get('region');
      Customer__c user = (Customer__c)testData.get('user');
      Customer__c manager = (Customer__c)testData.get('manager');
      Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.ALLUSERSHISTORY,region.Id,
manager.id));

      ViewAllUsersForAreaController viewAllUsersForAreaInstance = new ViewAllUsersForAreaController();
      List<SelectOption> users = viewAllUsersForAreaInstance.getUsers();

      //check user
      System.assertEquals(user.Id, users[0].getValue());
   }

   @isTest
   static void testBackReference(){
      Map<String, SObject> testData = TestDataSet.createTestData();
      Region__c region = (Region__c)testData.get('region');
      Customer__c manager = (Customer__c)testData.get('manager');
      Test.setCurrentPage(PageReferenceCreator.createReference(PathConst.ALLUSERSHISTORY, region.Id,
manager.id));

      ViewAllUsersForAreaController viewAllUsersForAreaInstance = new ViewAllUsersForAreaController();
      PageReference expectedPage = viewAllUsersForAreaInstance.back();

      //check url
      System.assertEquals('/apex/ManagerWorkflow', expectedPage.getUrl());
      //check manager
      System.assertEquals(manager.Id, expectedPage.getCookies().get('id').getValue());
   }
}

public class UpdateMetersController{
```

```
    public Customer__c user {get; set;}

    public UpdateMetersController(){
        String userId = PageReferenceCreator.getId();
        user = [SELECT Id, Name, Last_Name__c,
                    Gas_meters_reading__c, Electricity_meters_reading__c, Water_meters_reading__c
                FROM Customer__c
                WHERE id = :userId
                LIMIT 1];
    }

    public PageReference save(){
        update user;
        return PageReferenceCreator.createReference(PathConst.USERWORKFLOW, user.Id);
    }
}

public class UserWorkflowController {
    public List<Customer__c> users {get;set;}
    public Region__c region {get;set;}
    public Customer__c user {get;set;}
    public Payment__c paymentObject{get;set;}

    public Decimal gasDebt {get;set;}
    public Decimal electricityDebt {get;set;}
    public Decimal waterDebt {get;set;}

    public String gasRate {get;set;}


    public UserWorkflowController() {
        String userId = PageReferenceCreator.getId();
        users = [SELECT Id, Name, Email__c, Last_Name__c, Address__c, Address__r.Id, Address__r.Name,
Address__r.Region__r.Id,
                    Gas_meters_reading__c, Electricity_meters_reading__c, Water_meters_reading__c,
                    Summary_debt__c, Summary_debt_for_current_month__c,
                    (SELECT Name, Paid_Price__c, byManager__c
                    FROM Payments__r
                    WHERE byManager__c = false
                    ORDER BY CreatedDate DESC
                    LIMIT 1)
                FROM Customer__c
                WHERE Id = :userId
                LIMIT 1];

        region = [SELECT Name, Gas_rate__c, Electricity_rate__c, Water_rate__c
                FROM Region__c
                WHERE Id =:users.get(0).Address__r.Region__r.Id
                LIMIT 1];

        paymentObject = new Payment__c();

        if (!users.isEmpty()) {
            user = users.get(0);
        }

        if (!user.Payments__r.isEmpty()) {
            paymentObject = user.Payments__r[0];

        }

        debtInit();
```

```
            user.Summary_debt_for_current_month__c = UserWorkflowHelper.monthPaymentCalculator(gasDebt,
electricityDebt, waterDebt);
        }

        public PageReference updateUser(){
            update user;
            return null;
        }


        public PageReference toChangeAddress(){
            if(UserWorkflowHelper.isDebt(user.Summary_debt__c)){
                MessageThrower.addMessageError(WarningConst.ISDEBTSERROR);
                return null;
            }
            else{
                return PageReferenceCreator.createReference(PathConst.CHANGEADDRESS, user.Id);
            }
        }

        public void debtInit(){
            gasDebt = UserWorkflowHelper.debtCalculator(region.Gas_rate__c, user.Gas_meters_reading__c);
            electricityDebt = UserWorkflowHelper.debtCalculator(region.Electricity_rate__c,
user.Electricity_meters_reading__c);
            waterDebt = UserWorkflowHelper.debtCalculator(region.Water_rate__c, user.Water_meters_reading__c);

        }

        public PageReference toUpdateMeters(){
            return PageReferenceCreator.createReference(PathConst.UPDATEMETERS, user.Id);
        }

        public PageReference payment(){
            return PageReferenceCreator.createReference(PathConst.PAYMENT, user.Id);
        }

        public PageReference contactManager(){
            return PageReferenceCreator.createReference(PathConst.CONTACTMANAGER, user.Id);
        }

        public PageReference paymentHistory(){
            return PageReferenceCreator.createReference(PathConst.PAYMENTHISTORY, user.Id);
        }

    }

    public class UserWorkflowHelper{

        public static Decimal debtCalculator(Decimal rate, Decimal meters){
            Decimal debt = 0.00;
            if(meters == 0){
                MessageThrower.addMessageWarning(WarningConst.ZEROMETERSERROR);
                return debt;
            }
            else{
              debt = rate * meters;
              return debt;
            }

        }

        public static Decimal monthPaymentCalculator(Decimal gasDebt, Decimal electricityDebt, Decimal
waterDebt){
```

```
        Decimal monthlySummary = gasDebt + electricityDebt + waterDebt;
        return monthlySummary;
    }

    public static Boolean isDebt(Decimal summaryDebts){
        if (summaryDebts <= 0) {
            return false;
        }
        else{
            return true;
        }

    }

}

public class ViewAllUsersForAreaController{

    public List<Customer__c> regionUsers {get;set;}

    public ViewAllUsersForAreaController(){
        String regionId = PageReferenceCreator.getId();
        regionUsers = [SELECT Name, Last_Name__c
                    FROM Customer__c
                    WHERE Address__r.region__r.Id = :regionId
                    ORDER BY Name];

    }

    public List<SelectOption> getUsers(){
        return GetListOfUsers.listOfUsers(regionUsers);
    }

    public PageReference back(){
        String managerId = PageReferenceCreator.getParam('managerid');
        return PageReferenceCreator.createReference(PathConst.MANAGERWORKFLOW, managerId);
    }
}

public class WarningConst{
    public static final String ISDEBTSERROR = 'Pay your debts first!';
    public static final String ZEROMETERSERROR = 'Please, fill your monthly meters';
    public static final String LOGINERROR = 'Wrong login or password';
    public static final String USEREXIST = 'User with such email already exist!';
    public static final String INPUTERROR = 'Please, check your input';
    public static final String PASSWORDINPUTERROR = 'Password does not match';
    public static final String PAYMENTFAILED = 'Payment failed';
    public static final String EMPTYPRICE = 'Input payment amount';
    public static final String EMPTYEMAILTEXT = 'Fill your request first';
    public static final String RATESSAVED = 'New rates successfully saved!';
    public static final String MONTHLYPAYMENTUPDATED = 'Monthly Payment was updated';
    public static final String PERMISSIONERROR = 'You don\'t have permissions as manager';
}

<apex:page controller="AllUsersPaymentHistoryController" lightningStylesheets="true">
    <apex:pageMessages ></apex:pageMessages>
    <apex:form >
        <apex:pageBlock >
            <apex:selectList label="Choose user: " value="{!userId}" size="1" required="true">
                    <apex:selectOptions value="{!users}"/>
            </apex:selectList>
```

```
                <apex:pageBlockTable id="historyBlock" rendered="{!IF(selectedUser.Id = null,false,true)}"
value="{!payments}" var="payment">
                    <apex:column value="{!payment.CreatedDate}"/>
                    <apex:column value="{!payment.Name}"/>
                    <apex:column value="{!payment.Paid_Price__c}"/>
                    <apex:column value="{!payment.Remain_Debt__c}"/>
                    <apex:column value="{!payment.byManager__c}"/>
                </apex:pageBlockTable>
                <apex:pageBlockButtons location="top">
                    <apex:commandButton value="Apply user" action="{!rerender}" >
                        <apex:actionSupport event="onclick" reRender="historyBlock"/>
                    </apex:commandButton>
                    <apex:commandButton value="Back" action="{!back}"/>
                </apex:pageBlockButtons>
            </apex:pageBlock>
        </apex:form>
    </apex:page>

    <apex:page controller="BillController" renderAs="PDF">
        <apex:pageMessages ></apex:pageMessages>
        <apex:form >
            <apex:pageBlock title="Payment: {!dateOfPayment} - {!bill.User__r.Name}
{!bill.User__r.Last_Name__c}">
                <apex:pageBlockSection columns="1">
                    <apex:outputText label="Paid Price" value="{!bill.Paid_Price__c}"/>
                    <apex:outputText label="Remain debt" value="{!bill.Remain_Debt__c}"/>
                </apex:pageBlockSection>
            </apex:pageBlock>
        </apex:form>
    </apex:page>

    <apex:page controller="ChangeAddressController" lightningStylesheets="true">
        <apex:pageMessages ></apex:pageMessages>
        <apex:form >
            <apex:pageBlock title="{!user.Name} {!user.Last_Name__c}">
                <apex:pageBlockSection columns="1">
                    <apex:selectList label="Your current address: {!user.Address__r.Name}"
value="{!user.Address__c}" size="1" required="true">
                        <apex:selectOptions value="{!addresses}"/>
                    </apex:selectList>
                </apex:pageBlockSection>
                <apex:pageBlockButtons location="bottom">
                    <apex:commandButton value="Save" action="{!save}" />
                </apex:pageBlockButtons>
            </apex:pageBlock>
        </apex:form>
    </apex:page>

    <apex:page controller="ChangeMonthlyPaymentController" lightningStylesheets="true">
        <apex:pageMessages ></apex:pageMessages>
        <apex:form >
            <apex:pageBlock title="Change user's monthly payment">
                <apex:selectList label="Choose user: " value="{!userId}" size="1" required="true">
                    <apex:selectOptions value="{!users}"/>
                </apex:selectList>

                <apex:pageBlockSection id="monthlypayment" rendered="{!IF(selectedUser.Id = null,false,true)}">
                    <apex:inputText label="Customer's Summary" value="{!paidPrice}" />
                </apex:pageBlockSection>
                <apex:pageBlockButtons location="top">
                    <apex:commandButton value="Apply user" action="{!rerender}" >
                        <apex:actionSupport event="onclick" reRender="monthlypayment"/>
                    </apex:commandButton>
```

```
            <apex:commandButton value="Save changes" action="{!save}"/>
            <apex:commandButton value="Back" action="{!back}"/>
        </apex:pageBlockButtons>
    </apex:pageBlock>
  </apex:form>
</apex:page>

<apex:page controller="ChangeRatesController" lightningStylesheets="true">
  <apex:pageMessages ></apex:pageMessages>
  <apex:form >
    <apex:pageBlock title="Changing rates of {!region.Name} region!">
      <apex:pageBlockSection columns="3">
        <apex:inputField value="{!region.Gas_rate__c}" label="Gas rate"/>
        <apex:inputField value="{!region.Electricity_rate__c}" label="Electricity rate"/>
        <apex:inputField value="{!region.Water_rate__c}" label="Water rate"/>
      </apex:pageBlockSection>
      <apex:pageBlockButtons location="top">
        <apex:commandButton value="Apply" action="{!save}"/>
        <apex:commandButton value="Back" action="{!back}"/>
      </apex:pageBlockButtons>
    </apex:pageBlock>
  </apex:form>
</apex:page>

<apex:page controller="ContactManagerController" lightningStylesheets="true">
  <apex:pageMessages ></apex:pageMessages>
  <apex:form >
    <apex:pageBlock title="Your manager will get email with your request. Input your request bellow.">
      <apex:pageBlockSection columns="1">
        <apex:inputTextarea value="{!emailText}"/>
      </apex:pageBlockSection>
      <apex:pageBlockButtons >
        <apex:commandButton value="Send" action="{!sendToManager}"/>
        <apex:commandButton value="Cancel" action="{!cancel}"/>
      </apex:pageBlockButtons>
    </apex:pageBlock>
  </apex:form>
</apex:page>

<apex:page controller="LoginController" lightningStylesheets="true">

  <apex:pageMessages ></apex:pageMessages>
  <apex:form >
    <apex:pageBlock title="Welcome to communal service maintenance">
      <apex:pageBlockSection columns="1">

          <apex:inputText label="Login" value="{!login}"/>
          <apex:inputSecret label="Password" value="{!password}"/>

      </apex:pageBlockSection>
      <apex:pageBlockButtons location="bottom">
        <apex:commandButton value="Sign Up" action="{!register}"/>
        <apex:commandButton value="Log in" action="{!getCustomer}"/>
        <apex:commandButton value="Log in as manager" action="{!managerWorkflow}"/>
      </apex:pageBlockButtons>
    </apex:pageBlock>
  </apex:form>
</apex:page>

<apex:page controller="ManagerWorkflowController" lightningStylesheets="true">
  <apex:pageMessages ></apex:pageMessages>
  <apex:form >
```

```
                    <apex:pageBlock title="{!manager.Name} {!manager.Last_Name__c}, your region is
{!region.Name}">
                     <apex:pageBlockSection columns="3">

                    <apex:outputText value="Gas rate per coub: {!region.Gas_rate__c} €"/>
                    <apex:outputText value="Electricity rate per coub: {!region.Electricity_rate__c} €"/>
                    <apex:outputText value="Water rate per coub: {!region.Water_rate__c} €"/>



                </apex:pageBlockSection>
                <apex:pageBlockButtons location="top">
                    <apex:commandButton value="Change Region rates" action="{!changeRates}"/>
                    <apex:commandButton value="Change Monthly Payments" action="{!updateMonthlyPayment}"/>
                    <apex:commandButton value="Show users History" action="{!usersPaymentHistory}"/>
                    <apex:commandButton value="Show all users" action="{!viewAllUsers}"/>
                </apex:pageBlockButtons>
            </apex:pageBlock>
        </apex:form>
    </apex:page>

    <apex:page controller="PaymentHistoryController" lightningStylesheets="true">
    <apex:messages />
    <apex:form >
        <apex:pageBlock title="Check your payments {!user.Name} {!user.Last_Name__c}">
            <apex:pageBlockTable value="{!payments}" var="payment">
                <apex:column value="{!payment.CreatedDate}"/>
                <apex:column value="{!payment.Name}"/>
                <apex:column value="{!payment.Paid_Price__c}"/>
                <apex:column value="{!payment.Remain_Debt__c}"/>
                <apex:column value="{!payment.byManager__c}"/>
            </apex:pageBlockTable>
            <apex:pageBlockButtons location="top">
                <apex:commandButton value="Back" action="{!back}"/>
            </apex:pageBlockButtons>
        </apex:pageBlock>
    </apex:form>
    </apex:page>

    <apex:page controller="PaymentController" lightningStylesheets="true">
        <apex:pageMessages ></apex:pageMessages>
        <apex:form >
            <apex:pageBlock title="Pay your debts with comfort">
                <apex:pageBlockSection columns="1">

                    <apex:inputText label="Sum of payment" value="{!paidPrice}"/>

                </apex:pageBlockSection>

                <apex:pageBlockButtons >
                    <apex:commandButton value="Pay" action="{!createPayment}"/>
                    <apex:commandButton value="Cancel" action="{!cancel}"/>
                </apex:pageBlockButtons>
            </apex:pageBlock>
        </apex:form>
    </apex:page>

    <apex:page controller="RegisterController" lightningStylesheets="true">
        <apex:pageMessages ></apex:pageMessages>
        <apex:form >
            <apex:pageBlock title="Register your new personal account">
                <apex:pageBlockSection columns="2">
```

```
                    <apex:inputField label="Name: " value="{!user.Name}"/>
                    <apex:inputField label="Last Name: " value="{!user.Last_Name__c}"/>
                    <apex:inputField label="Email: " value="{!user.Email__c}"/>
                    <apex:selectList label="Choose your address: " value="{!user.Address__c}" size="1"
required="true">
                        <apex:selectOptions value="{!addresses}"/>
                    </apex:selectList>

                </apex:pageBlockSection>
                <apex:pageBlockSection columns="1">
                    <apex:inputSecret label="Account password: " value="{!user.Password__c}"/>
                    <apex:inputSecret label="Repeat your password" value="{!repeatPasswordInput}"/>
                </apex:pageBlockSection>

                <apex:pageBlockButtons location="bottom">
                    <apex:commandButton value="Create account" action="{!createNewUser}"/>
                </apex:pageBlockButtons>
            </apex:pageBlock>
        </apex:form>
    </apex:page>

    <apex:page controller="UpdateMetersController" lightningStylesheets="true">
        <apex:pageMessages ></apex:pageMessages>
        <apex:form >
            <apex:pageBlock title="{!user.Name} {!user.Last_Name__c}">
                <apex:pageBlockSection columns="3">
                    <apex:inputField required="true" label="Gas: " value="{!user.Gas_meters_reading__c}"/>
                    <apex:inputField required="true" label="Electricity: "
value="{!user.Electricity_meters_reading__c}"/>
                    <apex:inputField required="true" label="Water: " value="{!user.Water_meters_reading__c}"/>
                </apex:pageBlockSection>
                <apex:pageBlockButtons location="bottom">
                    <apex:commandButton value="Save" action="{!save}"/>
                </apex:pageBlockButtons>
            </apex:pageBlock>
        </apex:form>
    </apex:page>

    <apex:page controller="UserWorkflowController" action="{!updateUser}" lightningStylesheets="true">
        <apex:pageMessages ></apex:pageMessages>
        <apex:form >
            <apex:pageBlock title="{!user.Name} {!user.Last_Name__c}">

                <apex:pageBlockSection columns="3">
                    <apex:outputText label="Gas rate per coub: " value="{!region.Gas_rate__c} €"/>
                    <apex:outputText label="Electricity rate per coub: " value="{!region.Electricity_rate__c} €"/>
                    <apex:outputText label="Water rate per coub: " value="{!region.Water_rate__c} €"/>


                    <apex:outputText label="Your gas readings: " value="{!user.Gas_meters_reading__c}"/>
                    <apex:outputText label="Your electricity readings: " value="{!user.Electricity_meters_reading__c}"/>
                    <apex:outputText label="Your water readings: " value="{!user.Water_meters_reading__c}"/>


                    <apex:outputText label="Your gas debt: " value="{!gasDebt} €"/>
                    <apex:outputText label="Your electricity debt: " value="{!electricityDebt} €"/>
                    <apex:outputText label="Your water debt: " value="{!waterDebt} €"/>


                    <apex:outputText label="Address: " value="{!user.Address__r.Name}"/>
                    <apex:outputText label="Region: " value="{!region.Name}"/>
                    <apex:outputText label="Your summary debt for that month: "
value="{!IF(user.Summary_debt_for_current_month__c = null, 0.00, user.Summary_debt_for_current_month__c)}"/>
```

```
            <apex:outputText label="Last time you pay in " value="{!paymentObject.Name}"/>
            <apex:outputText label="You paid: " value="{!paymentObject.Paid_Price__c}"/>
            <apex:outputText label="Your summary debt for all time: " value="{!user.Summary_debt__c}"/>

        </apex:pageBlockSection>

        <apex:pageBlockButtons location="top">

            <apex:commandButton value="Update Your Meters" action="{!toUpdateMeters}"/>
            <apex:commandButton value="Change your address" action="{!toChangeAddress}"/>
            <apex:commandButton value="Payment" action="{!payment}"/>
            <apex:commandButton value="Contact your manager" action="{!contactManager}"/>
            <apex:commandButton value="Payment history" action="{!paymentHistory}"/>
        </apex:pageBlockButtons>



    </apex:pageBlock>
  </apex:form>
</apex:page>

<apex:page controller="ViewAllUsersForAreaController" lightningStylesheets="true">
  <apex:pageMessages ></apex:pageMessages>
  <apex:form >
    <apex:pageBlock title="View all your users below">
      <apex:pageBlockTable value="{!regionUsers}" var="user">
        <apex:column value="{!user.Name} {!user.Last_Name__c}"/>
      </apex:pageBlockTable>
      <apex:pageBlockButtons location="top">
        <apex:commandButton value="Back" action="{!back}"/>
      </apex:pageBlockButtons>
    </apex:pageBlock>
  </apex:form>
</apex:page>

<?xml version="1.0" encoding="UTF-8"?>
<CustomSite xmlns="http://soap.sforce.com/2006/04/metadata">
  <active>true</active>
  <allowHomePage>false</allowHomePage>
  <allowStandardAnswersPages>false</allowStandardAnswersPages>
  <allowStandardIdeasPages>false</allowStandardIdeasPages>
  <allowStandardLookups>false</allowStandardLookups>
  <allowStandardPortalPages>true</allowStandardPortalPages>
  <allowStandardSearch>false</allowStandardSearch>
  <authorizationRequiredPage>Unauthorized</authorizationRequiredPage>
  <bandwidthExceededPage>BandwidthExceeded</bandwidthExceededPage>
  <browserXssProtection>true</browserXssProtection>
  <clickjackProtectionLevel>AllowAllFraming</clickjackProtectionLevel>
  <contentSniffingProtection>true</contentSniffingProtection>
  <cspUpgradeInsecureRequests>true</cspUpgradeInsecureRequests>
  <description>page for managing your communal services</description>
  <enableAuraRequests>true</enableAuraRequests>
  <fileNotFoundPage>FileNotFound</fileNotFoundPage>
  <genericErrorPage>Exception</genericErrorPage>
  <inMaintenancePage>InMaintenance</inMaintenancePage>
  <inactiveIndexPage>InMaintenance</inactiveIndexPage>
  <indexPage>LoginPage</indexPage>
  <masterLabel>Easy To Pay</masterLabel>
  <referrerPolicyOriginWhenCrossOrigin>true</referrerPolicyOriginWhenCrossOrigin>
  <requireHttps>true</requireHttps>
  <requireInsecurePortalAccess>false</requireInsecurePortalAccess>
```

```
    <siteAdmin>oleksii.ilin@alexinc.com</siteAdmin>
    <siteGuestRecordDefaultOwner>oleksii.ilin@alexinc.com</siteGuestRecordDefaultOwner>
    <siteTemplate>SiteTemplate</siteTemplate>
    <siteType>Visualforce</siteType>
    <subdomain>payyourservice-developer-edition</subdomain>
</CustomSite>

trigger LimitDebtTrigger on Customer__c (after update, after insert) {
    if (Trigger.new.isEmpty()==false) {
        for (Customer__c user : Trigger.new) {
            if (user.Summary_debt__c >= 7500.00) {

                Address__c address = [SELECT Region__c,
                                      (SELECT Email__c
                                      FROM Customers__r
                                      WHERE isManager__c = true
                                      LIMIT 1)
                                  FROM Address__c
                                  WHERE Id = :user.Address__c
                                  LIMIT 1];

                String managerEmail = address.Customers__r[0].Email__c;

                Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
                email.toAddresses = new String[]{managerEmail};
                email.setSubject('Debt limit has reached');
                email.optOutPolicy = 'FILTER';

                email.plainTextBody = user.Name+' '+ user.Last_Name__c + ' has reached his/her debt limit\n'+
                'Hi! Now his/her summary debt is: '+user.Summary_debt__c+' €\n'+
                'You must make him/her pay as fast as it possible or we will find him/her... They won\'t like that.\n'+
                'Use that email to contact customer '+user.Email__c+'\n'+
                'Best regards,\n'+
                'Evil corporation';

                Messaging.SingleEmailMessage[] messages = new List<Messaging.SingleEmailMessage> {email};
                Messaging.sendEmail(messages);
            }
        }
    }

}

<?xml version="1.0" encoding="UTF-8"?>
<Workflow xmlns="http://soap.sforce.com/2006/04/metadata">
    <alerts>
        <fullName>Limit_Email</fullName>
        <description>Send when debt limit reached</description>
        <protected>false</protected>
        <recipients>
            <field>Email__c</field>
            <type>email</type>
        </recipients>
        <senderType>CurrentUser</senderType>
        <template>unfiled$public/Limit_Template</template>
    </alerts>
</Workflow>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Region__c</fullName>
    <deleteConstraint>SetNull</deleteConstraint>
    <externalId>false</externalId>
```

```
    <label>Region</label>
    <referenceTo>Region__c</referenceTo>
    <relationshipLabel>Addresses</relationshipLabel>
    <relationshipName>Addresses</relationshipName>
    <required>false</required>
    <trackTrending>false</trackTrending>
    <type>Lookup</type>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>Accept</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Accept</actionName>
        <formFactor>Large</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Accept</actionName>
        <formFactor>Small</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>CancelEdit</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>CancelEdit</actionName>
        <formFactor>Large</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>CancelEdit</actionName>
        <formFactor>Small</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Clone</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Clone</actionName>
        <formFactor>Large</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Clone</actionName>
        <formFactor>Small</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Delete</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Delete</actionName>
        <formFactor>Large</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
```

```
    <actionName>Delete</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Edit</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Edit</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Edit</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>List</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>List</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>List</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>New</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>New</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>New</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>SaveEdit</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>SaveEdit</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>SaveEdit</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Tab</actionName>
    <type>Default</type>
```

```
    </actionOverrides>
    <actionOverrides>
        <actionName>Tab</actionName>
        <formFactor>Large</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Tab</actionName>
        <formFactor>Small</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>View</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>View</actionName>
        <formFactor>Large</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>View</actionName>
        <formFactor>Small</formFactor>
        <type>Default</type>
    </actionOverrides>
    <allowInChatterGroups>false</allowInChatterGroups>
    <compactLayoutAssignment>SYSTEM</compactLayoutAssignment>
    <deploymentStatus>Deployed</deploymentStatus>
    <enableActivities>false</enableActivities>
    <enableBulkApi>true</enableBulkApi>
    <enableFeeds>false</enableFeeds>
    <enableHistory>false</enableHistory>
    <enableLicensing>false</enableLicensing>
    <enableReports>false</enableReports>
    <enableSearch>true</enableSearch>
    <enableSharing>true</enableSharing>
    <enableStreamingApi>true</enableStreamingApi>
    <label>Address</label>
    <nameField>
        <label>Address Name</label>
        <type>Text</type>
    </nameField>
    <pluralLabel>Addresses</pluralLabel>
    <searchLayouts/>
    <sharingModel>ReadWrite</sharingModel>
    <visibility>Public</visibility>
</CustomObject>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Address__c</fullName>
    <deleteConstraint>Restrict</deleteConstraint>
    <externalId>false</externalId>
    <label>Address</label>
    <referenceTo>Address__c</referenceTo>
    <relationshipLabel>Customers</relationshipLabel>
    <relationshipName>Customers</relationshipName>
    <required>true</required>
    <trackTrending>false</trackTrending>
    <type>Lookup</type>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
```

```
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Electricity_meters_reading__c</fullName>
    <defaultValue>0</defaultValue>
    <externalId>false</externalId>
    <label>Electricity meters reading</label>
    <precision>18</precision>
    <required>false</required>
    <scale>2</scale>
    <trackTrending>false</trackTrending>
    <type>Number</type>
    <unique>false</unique>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Email__c</fullName>
    <caseSensitive>true</caseSensitive>
    <externalId>true</externalId>
    <label>Email</label>
    <required>true</required>
    <trackTrending>false</trackTrending>
    <type>Email</type>
    <unique>true</unique>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Gas_meters_reading__c</fullName>
    <defaultValue>0</defaultValue>
    <externalId>false</externalId>
    <label>Gas meters reading</label>
    <precision>18</precision>
    <required>false</required>
    <scale>2</scale>
    <trackTrending>false</trackTrending>
    <type>Number</type>
    <unique>false</unique>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>isManager__c</fullName>
    <defaultValue>false</defaultValue>
    <externalId>false</externalId>
    <label>isManager</label>
    <trackTrending>false</trackTrending>
    <type>Checkbox</type>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Last_Name__c</fullName>
    <externalId>false</externalId>
    <label>Last Name</label>
    <length>80</length>
    <required>true</required>
    <trackTrending>false</trackTrending>
    <type>Text</type>
    <unique>false</unique>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
```

```
    <fullName>Password__c</fullName>
    <externalId>false</externalId>
    <label>Password</label>
    <length>250</length>
    <required>true</required>
    <trackTrending>false</trackTrending>
    <type>Text</type>
    <unique>false</unique>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Phone__c</fullName>
    <externalId>false</externalId>
    <label>Phone</label>
    <required>false</required>
    <trackTrending>false</trackTrending>
    <type>Phone</type>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Summary_debt__c</fullName>
    <externalId>false</externalId>
    <formula>Summary_debt_for_current_month__c</formula>
    <formulaTreatBlanksAs>BlankAsZero</formulaTreatBlanksAs>
    <label>Summary debt</label>
    <precision>18</precision>
    <required>false</required>
    <scale>2</scale>
    <trackTrending>false</trackTrending>
    <type>Currency</type>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Summary_debt_for_current_month__c</fullName>
    <externalId>false</externalId>
    <label>Summary debt for current month</label>
    <precision>18</precision>
    <required>false</required>
    <scale>2</scale>
    <trackTrending>false</trackTrending>
    <type>Currency</type>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Water_meters_reading__c</fullName>
    <defaultValue>0</defaultValue>
    <externalId>false</externalId>
    <label>Water meters reading</label>
    <precision>18</precision>
    <required>false</required>
    <scale>2</scale>
    <trackTrending>false</trackTrending>
    <type>Number</type>
    <unique>false</unique>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
```

```
    <actionName>Accept</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Accept</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Accept</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>CancelEdit</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>CancelEdit</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>CancelEdit</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Clone</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Clone</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Clone</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Delete</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Delete</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Delete</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Edit</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Edit</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
```

```
</actionOverrides>
<actionOverrides>
    <actionName>Edit</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>List</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>List</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>List</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>New</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>New</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>New</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>SaveEdit</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>SaveEdit</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>SaveEdit</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Tab</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Tab</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Tab</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
```

```
        <actionName>View</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>View</actionName>
        <formFactor>Large</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>View</actionName>
        <formFactor>Small</formFactor>
        <type>Default</type>
    </actionOverrides>
    <allowInChatterGroups>false</allowInChatterGroups>
    <compactLayoutAssignment>SYSTEM</compactLayoutAssignment>
    <deploymentStatus>Deployed</deploymentStatus>
    <enableActivities>false</enableActivities>
    <enableBulkApi>true</enableBulkApi>
    <enableFeeds>false</enableFeeds>
    <enableHistory>false</enableHistory>
    <enableLicensing>false</enableLicensing>
    <enableReports>false</enableReports>
    <enableSearch>true</enableSearch>
    <enableSharing>true</enableSharing>
    <enableStreamingApi>true</enableStreamingApi>
    <label>Customer</label>
    <nameField>
        <label>First Name</label>
        <type>Text</type>
    </nameField>
    <pluralLabel>Customers</pluralLabel>
    <searchLayouts/>
    <sharingModel>ReadWrite</sharingModel>
    <visibility>Public</visibility>
</CustomObject>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>byManager__c</fullName>
    <defaultValue>false</defaultValue>
    <description>That field mean, that payment provided by manager.</description>
    <externalId>false</externalId>
    <label>byManager</label>
    <trackTrending>false</trackTrending>
    <type>Checkbox</type>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Paid_Price__c</fullName>
    <externalId>false</externalId>
    <label>Paid Price</label>
    <precision>18</precision>
    <required>true</required>
    <scale>2</scale>
    <trackTrending>false</trackTrending>
    <type>Currency</type>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Remain_Debt__c</fullName>
    <externalId>false</externalId>
```

```
    <formula>User__r.Summary_debt_for_current_month__c  - Paid_Price__c</formula>
    <formulaTreatBlanksAs>BlankAsZero</formulaTreatBlanksAs>
    <label>Remain Debt</label>
    <precision>18</precision>
    <required>false</required>
    <scale>2</scale>
    <trackTrending>false</trackTrending>
    <type>Currency</type>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>User__c</fullName>
    <deleteConstraint>SetNull</deleteConstraint>
    <externalId>false</externalId>
    <label>User</label>
    <referenceTo>Customer__c</referenceTo>
    <relationshipLabel>Payments</relationshipLabel>
    <relationshipName>Payments</relationshipName>
    <required>false</required>
    <trackTrending>false</trackTrending>
    <type>Lookup</type>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>Accept</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Accept</actionName>
        <formFactor>Large</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Accept</actionName>
        <formFactor>Small</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>CancelEdit</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>CancelEdit</actionName>
        <formFactor>Large</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>CancelEdit</actionName>
        <formFactor>Small</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Clone</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Clone</actionName>
        <formFactor>Large</formFactor>
        <type>Default</type>
    </actionOverrides>
```

```
<actionOverrides>
    <actionName>Clone</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Delete</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Delete</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Delete</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Edit</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Edit</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Edit</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>List</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>List</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>List</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>New</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>New</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>New</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>SaveEdit</actionName>
```

```
    <type>Default</type>
  </actionOverrides>
  <actionOverrides>
    <actionName>SaveEdit</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
  </actionOverrides>
  <actionOverrides>
    <actionName>SaveEdit</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
  </actionOverrides>
  <actionOverrides>
    <actionName>Tab</actionName>
    <type>Default</type>
  </actionOverrides>
  <actionOverrides>
    <actionName>Tab</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
  </actionOverrides>
  <actionOverrides>
    <actionName>Tab</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
  </actionOverrides>
  <actionOverrides>
    <actionName>View</actionName>
    <type>Default</type>
  </actionOverrides>
  <actionOverrides>
    <actionName>View</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
  </actionOverrides>
  <actionOverrides>
    <actionName>View</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
  </actionOverrides>
  <allowInChatterGroups>false</allowInChatterGroups>
  <compactLayoutAssignment>SYSTEM</compactLayoutAssignment>
  <deploymentStatus>Deployed</deploymentStatus>
  <enableActivities>false</enableActivities>
  <enableBulkApi>true</enableBulkApi>
  <enableFeeds>false</enableFeeds>
  <enableHistory>false</enableHistory>
  <enableLicensing>false</enableLicensing>
  <enableReports>false</enableReports>
  <enableSearch>true</enableSearch>
  <enableSharing>true</enableSharing>
  <enableStreamingApi>true</enableStreamingApi>
  <label>Payment</label>
  <nameField>
    <displayFormat>{YYYY}{MM}-{0}</displayFormat>
    <label>Payment Name</label>
    <type>AutoNumber</type>
  </nameField>
  <pluralLabel>Payments</pluralLabel>
  <searchLayouts/>
  <sharingModel>ReadWrite</sharingModel>
  <visibility>Public</visibility>
</CustomObject>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Electricity_rate__c</fullName>
    <externalId>false</externalId>
    <label>Electricity rate</label>
    <precision>18</precision>
    <required>true</required>
    <scale>2</scale>
    <trackTrending>false</trackTrending>
    <type>Currency</type>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Gas_rate__c</fullName>
    <externalId>false</externalId>
    <label>Gas rate</label>
    <precision>18</precision>
    <required>true</required>
    <scale>2</scale>
    <trackTrending>false</trackTrending>
    <type>Currency</type>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomField xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>Water_rate__c</fullName>
    <externalId>false</externalId>
    <label>Water rate</label>
    <precision>18</precision>
    <required>true</required>
    <scale>2</scale>
    <trackTrending>false</trackTrending>
    <type>Currency</type>
</CustomField>

<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>Accept</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Accept</actionName>
        <formFactor>Large</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Accept</actionName>
        <formFactor>Small</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>CancelEdit</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>CancelEdit</actionName>
        <formFactor>Large</formFactor>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
```

```
    <actionName>CancelEdit</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Clone</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Clone</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Clone</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Delete</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Delete</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Delete</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Edit</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Edit</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Edit</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>List</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>List</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>List</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>New</actionName>
    <type>Default</type>
```

```
</actionOverrides>
<actionOverrides>
    <actionName>New</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>New</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>SaveEdit</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>SaveEdit</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>SaveEdit</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Tab</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Tab</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>Tab</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>View</actionName>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>View</actionName>
    <formFactor>Large</formFactor>
    <type>Default</type>
</actionOverrides>
<actionOverrides>
    <actionName>View</actionName>
    <formFactor>Small</formFactor>
    <type>Default</type>
</actionOverrides>
<allowInChatterGroups>false</allowInChatterGroups>
<compactLayoutAssignment>SYSTEM</compactLayoutAssignment>
<deploymentStatus>Deployed</deploymentStatus>
<enableActivities>false</enableActivities>
<enableBulkApi>true</enableBulkApi>
<enableFeeds>false</enableFeeds>
<enableHistory>false</enableHistory>
<enableLicensing>false</enableLicensing>
<enableReports>false</enableReports>
<enableSearch>true</enableSearch>
```

```
    <enableSharing>true</enableSharing>
    <enableStreamingApi>true</enableStreamingApi>
    <label>Region</label>
    <nameField>
      <label>Region Name</label>
      <type>Text</type>
    </nameField>
    <pluralLabel>Regions</pluralLabel>
    <searchLayouts/>
    <sharingModel>ReadWrite</sharingModel>
    <visibility>Public</visibility>
</CustomObject>

<?xml version="1.0" encoding="UTF-8"?>
<ConnectedApp xmlns="http://soap.sforce.com/2006/04/metadata">
    <contactEmail>oleksii.llin@sparkybit.com</contactEmail>
    <label>Payment Access</label>
    <oauthConfig>
      <callbackUrl>https://access_token/oauth/success</callbackUrl>

<consumerKey>3MVG91BJr_0ZDQ4tV15NHF357DzlGSBl3Gu0fcthIPZbkjvXnb24sV7VIUGzdbzWgfalcMpxwXR.P
dqLwvXHg</consumerKey>
      <isAdminApproved>false</isAdminApproved>
      <scopes>Api</scopes>
      <scopes>Web</scopes>
      <scopes>Full</scopes>
      <scopes>CustomApplications</scopes>
      <scopes>RefreshToken</scopes>
    </oauthConfig>
</ConnectedApp>
```

У цьому додатку були вказали файли з кожного розділу файлової системи. Повний перелік файлів зі змістом знаходяться на електронному носієві.

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра
на тему:
«Розробка веб-орієнтованої інформаційної системи обліку надання комунальних послуг з використанням Salesforce»
студента групи 122-18ск-2 Ільїна Олексія Володимировича

Керівник економічного розділу
доцент каф. ПЕП та ПУ, к.е.н **О.Г. Вагонова**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Ільїн.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Ільїн.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
diplom.zip	Архів. Містить коди програми.
Презентація	
Презентація Ільїн.ppt	Презентація кваліфікаційної роботи.