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

Навчально-науковий
інститут електроенергетики
(інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп’ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
Кваліфікаційної роботи ступеня магістра

студента Билбаса Костянтина Олександровича
(ПІБ)
академічної групи 123м-22-1
(шифр)
спеціальності 123 «Комп’ютерна інженерія»
(код і назва спеціальності)
за освітньо-професійною програмою «Комп’ютерна інженерія»
(офіційна назва)

на тему «Обґрунтування структури комп'ютерної системи ІТ-компанії з урахуванням сервісу контроля розсилки електронних листів»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Сергєєва К.Л.			
розділів:				
синтез системи	доц. Бешта Д.О.			
розроблення програмного забезпечення	ас. Панферова Я.В.			

Рецензент				

Нормоконтролер	доц. Шедловська Я.І.			

Дніпро
2023

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)
«06» вересня 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня магістра
(бакалавра, спеціаліста, магістра)

студенту Билбасу К.О. **академічної групи** 123м-22-1
(прізвище та ініціали) (шифр)
спеціальності 123 «Комп'ютерна інженерія»
за освітньо-професійною програмою 123 «Комп'ютерна інженерія»
(офіційна назва)

на тему «Обґрунтування структури комп'ютерної системи ІТ-компанії з урахуванням сервісу контроля розсилки електронних листів», затверджену наказом ректора НТУ «Дніпровська політехніка» від 09.10.2023 р. № 1227-с

Розділ	Зміст завдання	Термін виконання
Стан питання та постановка завдання	На основі матеріалів практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизувати предмет та мету досліджень	10.10.2023
Теоретичний	Обґрунтувати теоретичну базу розв'язання наукового завдання, якому присвячено роботу	25.10.2023
Синтез системи	Розробка комп'ютерної системи	15.11.2023
Розроблення програмного забезпечення	Розробка програмного забезпечення	29.11.2023
Експериментальний розділ	Проведення і обробка результатів експериментів	30.11.2023

Завдання видано _____ доц. Сергєєва К.Л.
(підпис керівника) (прізвище, ініціали)
Дата видачі 06 вересня 2023 р.
Дата подання до екзаменаційної комісії 19.12.2023 р.
Прийнято до виконання _____ Билбас К.О.

РЕФЕРАТ

Пояснювальна записка містить 97 сторінок, 10 рисунків, 1 таблицю, 22 джерела переліку посилань.

Об’єкт розробки: комп’ютерна система контролю розсилки електронних листів.

Мета роботи: розробка комп’ютерної системи контролю автоматизованої розсилки електронних листів. Обґрунтування використання власної системи контролю розсилки поміж існуючих рішень.

Пояснювальна записки має аналіз існуючих систем контролю автоматизованої розсилки електронних повідомлень та описує недоліки та переваги кожної з них.

Середовище розробки:

- React.js, JavaScript, SCSS, PHP;
- VS Code, Figma, Photoshop, XAMPP.

В результаті роботи представлено веб-додаток, розроблений з урахуванням сучасного зручного дизайну та використанням сучасних технологій, з можливістю легкого масштабування функціоналу.

REACT, HTML, PHP, ВЕБ-САЙТ, SCSS, JAVASCRIPT, КОМП’ЮТЕРНА СИСТЕМА, ТАБЛИЦЯ, ТЕХНОЛОГІЇ, ІНФОРМАЦІЯ, ГЛОБАЛЬНА МЕРЕЖА, ЕЛЕКТРОННИЙ ЛИСТ, РОЗСИЛКА

ЗМІСТ

РЕФЕРАТ 3
ЗМІСТ 4
ВСТУП 6
1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ 7
1.1 Інформаційні технології в суспільстві та світі 7
1.2 Email-маркетинг 9
1.3 Плюси та мінуси email-маркетингу 12
1.4 Огляд існуючих систем керування розсилкою 13
2 ТЕОРЕТИЧНИЙ РОЗДІЛ 16
2.1 Вибір середовища програмування 16
2.2 Середовище Visual Studio Code 17
2.3 Вибір фреймворку для розробки клієнтської частини 18
2.4 React.js 20
2.5 Вибір мови для розробки серверної частини 23
2.6 PHP 25
3 СИНТЕЗ СИСТЕМИ РОЗСИЛКИ ЕЛЕКТРОННИХ ЛИСТІВ 27
3.1 Формулювання технічних вимог до системи керування розсилкою електронних листів 27
3.2 Формулювання технічних вимог до обладнання 28
3.2.1 Вимоги до серверного обладнання 28
3.2.2 Вимоги до мережевого обладнання 28
3.2.3 Вимоги до серверного програмного забезпечення 29
3.3 Вибір та обґрунтування застосування апаратних засобів 29
3.4 Синтез схеми системи 29
3.4.1 Синтез програмної схеми системи 29
3.4.2 Синтез апаратної схеми системи 31
4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ 33

4.1 Призначення й область застосування програмного забезпечення 33
4.1.1 Призначення 33
4.1.2 Область застосування 33
4.2 Обгрунтування технічних характеристик програм 34
4.2.1. Visual Studio Code (версія 1.84.2) 34
4.2.2. Node.js (версія 18.15.0) 34
4.2.3. Xampp (версія 8.1) 34
4.3 Опис розробленої програми 35
4.3.1 Розробка структурної схеми 35
4.3.2 Розробка дизайну 36
4.3.3 Розробка клієнтської частини 37
4.3.4 Розробка серверної частини 42
4.4 Очікувані техніко-економічні показники 45
5 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ 47
5.1 Мета і завдання експерименту 47
5.2 Вимоги до експерименту 48
5.3 Результати експерименту 49
5.3.1 Функціональність та Інтерфейс 49
5.3.2 Персоналізація 49
5.3.3 Безпека Даних 50
5.3.4 Оптимізація та Швидкодія 50
5.3.5 Вимоги до експерименту 50
ВИСНОВКИ 52
ПЕРЕЛІК ПОСИЛАНЬ 54
ДОДАТОК А 55
ДОДАТОК Б 86

ВСТУП

Створення нових комп'ютерних технологій має велике значення для розвитку суспільства.

Сьогодні інформаційні технології відіграють важливу роль у нашому житті. Використання комп'ютерів стало звичним явищем, хоча ще не так давно ситуація була інакшою. Інформаційні технології зробили наше життя більш різнобічним, а також полегшили більшість повсякденних задач. Сьогоднішнє суспільство важко уявити без інформаційних технологій.

Сьогодні email-маркетинг є важливою частиною будь-якої стратегії просування бізнесу в Інтернеті. Згідно з дослідженням Holistic Email Marketing та GetResponse, email-кампанії мають найвищу рентабельність інвестицій порівняно з іншими маркетинговими інструментами.

За ступенем конверсійності з email-маркетингом може змагатися тільки контекстна реклама. Однак класичні масові розсилки, які робили вручну, відходять у минуле. Бомбардування листами більше не працює. На перший план виходять сценарії email-розсилок - точкові, персоналізовані листи в потрібний момент часу. Для цього необхідна автоматизація email-розсилок.

Задля досягнення поставленої мети кваліфікаційної роботи необхідно вирішити наступні завдання:

- ♣ дослідити існуючи системи керування розсилкою електронних листів;
- ♣ дослідити потреби користувачів;
- ♣ розробити алгоритми керування даними та відправкою листів;
- ♣ програмно реалізувати розроблені методи і алгоритми.

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Інформаційні технології в суспільстві та світі

Сучасний етап життя світу характеризується визначальною роллю інформації та заснованої на ній економіки. Розвиток інформаційного суспільства прямо пов'язаний із потребою збирати, обробляти та передавати великі обсяги інформації, а також перетворювати її на товар із значною вартістю. Цей процес призвів до глобального переходу від індустріального суспільства до інформаційного. З'явлення всесвітньої мережі Інтернет суттєво збільшило міжнародну комунікацію в різних сферах людського життя.

Інформація вважається одним із найцінніших ресурсів суспільства поряд із традиційними матеріальними ресурсами, такими як нафта, метали, корисні копалини та інші. Таким чином, обробка інформації може бути розглянута як технологічний процес, аналогічний обробці матеріальних ресурсів. Інформаційна технологія передбачає вміння ефективно працювати з інформацією та комп'ютерами.

Роль інформаційних технологій в освіті та наукових дослідженнях надзвичайно важлива. Сьогодні важко уявити навчальний заклад без комп'ютерних класів та доступу до різноманітних електронних бібліотек, що дозволяє отримувати знання, не виходячи з дому, і значно спрощує процес навчання та самонавчання. В той же час інформаційні технології сприяють розвитку наукових знань.

Зі збільшенням швидкості обміну інформацією, з'являється можливість виконувати надскладні арифметичні розрахунки майже миттєво та багато іншого. Інформаційні технології стали одним із сучасних засобів комунікації, головною перевагою якого є його загальна доступність. За допомогою інформаційних технологій можна легко отримувати доступ до цікавої

інформації та взаємодіяти з реальними людьми. З одного боку, це може мати негативний вплив, оскільки спілкування "наживо" зменшується, але, з іншого боку, це відкриває можливість спілкування з людьми з будь-якого кутка світу, що має величезне значення.

Виробництво і транспорт, банківські та фондові установи, засоби масової інформації та видавництва, системи оборони, соціальні та правоохоронні бази даних, обслуговування та охорона здоров'я, процеси навчання, офіси для обробки наукової та ділової інформації, і, нарешті, Інтернет – всюди присутні інформаційні технології. Інформаційна насиченість не лише трансформувала світ, але й викликала нові проблеми, які не були передбачені.

З розвитком сучасних технологій збільшується прозорість світу, а також швидкість та обсяг передачі інформації між складовими світової системи, що створює додатковий фактор інтеграції. Це означає, що традиційна роль місцевих елементів, сприяючих самостійному інерційному розвитку, послаблюється. У той же час зростає реакція елементів на сигнали з позитивним зворотним зв'язком. Інтеграція може бути позитивною, але важливо уникати затухання регіональних та культурно-історичних особливостей розвитку.

За останні кілька років соціальні мережі стали найпопулярнішими ресурсами в Інтернеті: на сьогоднішній день Facebook, Twitter і Linkedin у США та Західній Європі є сайтами з мільйонами активних користувачів. Ці ресурси використовують 75% українських користувачів Інтернету.

Соціальні мережі справді заповнили наше життя, і це сталося завдяки швидкому розвитку інформаційних технологій взагалі.

Зараз важко знайти галузь, де не використовуються інформаційні технології. Підводячи підсумок, можна сказати, що інформаційні технології глибоко проникли в наше життя і сучасне суспільство, яке не може існувати без них в нинішньому вигляд

1.2 Email-маркетинг

Електронний маркетинг за допомогою електронної пошти є зручним і доступним методом розповсюдження різноманітної інформації, включаючи комерційну, серед вашої цільової аудиторії шляхом надсилання електронних листів. Це не спам, оскільки аудиторія, яка отримує повідомлення, попередньо підписана на розсилку, зацікавлена в отриманні пропозиції та має можливість відписатися в будь-який момент.

Email-маркетинг — це один із перших інструментів продажів, що з'явилися в Інтернеті. Незважаючи на те, що сьогодні з'явилася велика кількість нових каналів реклами, email-маркетинг міцно залишається на позиціях лідера у сфері інтернет-продажів.

Перший маркетинговий електронний лист було надіслано 1978 року, він приніс продажів на 13 мільйонів доларів США. З цього почалося використання каналу маркетингу, який і донині залишається одним із найпопулярніших. Незважаючи на наявність нових і більш інтерактивних каналів, таких як соціальні мережі та чати, електронна пошта продовжує залишатися ефективним способом формування власної аудиторії та невід'ємною частиною ефективних стратегій inbound-маркетингу.

Продажі через електронну пошту мають низку переваг, якими не можуть похвалитися альтернативні джерела лідів:

- Рентабельність інвестицій: грамотно складена email-кампанія окупає витрати на її підготовку, збільшує продажі та лояльність цільової аудиторії.
- Охоплення: порівняно з соц.мережами, де охоплення становить лише кілька відсотків, електронні листи отримують принаймні 90% бази підписників.

- Глибина читання: у соціальних мережах люди прокручують стрічку, майже не затримуючись на контенті. Крім того, після прочитання пост безслідно губиться. Електронний лист завжди залишається доступним на пошті, а на його прочитання більше половини користувачів витрачають понад 8 секунд.
- Дизайн і функції: у чатах і соц.мережах ми змушені користуватися тільки встановленими можливостями. Верстка email-розсилок має необмежену кількість варіантів дизайну. У лист можна додати будь-які інтерактивні елементи.
- Аналітика: аналітика месенджерів і соціальних мереж все ще залишається досить скромною. Зі свого боку, email-маркетинг дає змогу в деталях відстежувати поведінку користувачів.
- Конверсійність: ефективний email-маркетинг безпосередньо взаємодіє з цільовою аудиторією, яка зацікавлена в рекламних пропозиціях. Тому він забезпечує істотно вищий коефіцієнт конверсії, ніж інші канали, змагаючись тільки з контекстною рекламою в пошукових мережах. Так, у середньому відсоток конверсії від емейл-розсилок у 2 рази вищий, ніж з органічного трафіку на сайт, і в 9 разів вищий, ніж від трафіку з соціальних мереж.
- Аудиторія: кожен користувач Інтернету має активну електронну пошту, тому потенційно email-маркетинг дає змогу донести інформацію до багатьох користувачів мережі.

Це неповний список причин, навіщо потрібен email-маркетинг. На відміну від нових каналів, які тільки досліджуються і тестуються, масові розсилки давно пройшли випробування часом і мають у своєму розпорядженні перевірені технології продажів. Крім того, цей канал має кілька важливих переваг.

Низька вартість

Зрозуміло, що потрібні певні інвестиції, адже за масові розсилки потрібно платити за тарифами поштових сервісів. Але порівняно з більшістю інших рекламних каналів ці витрати не є суттєвими.

Персональний підхід

В email-маркетингу ви можете сегментувати і кастомізувати листи, щоб користувачі отримували тільки ті листи, які їх цікавлять. Наприклад, ви можете надсилати листи з пропозиціями купити аксесуари всім клієнтам, які придбали мобільні телефони. Існує висока ймовірність того, що вони скористаються цією пропозицією.

Легкий старт

Щоб розпочати роботу з email-маркетингом, потрібно не так вже й багато: створити форму підписки на розсилку, вибрати сервіс з інтуїтивно зрозумілим інтерфейсом і відправити перший лист до вашої бази даних. Автоматизація email-маркетингу значно спростила всі процеси. Головне — зробити контент корисним для одержувачів, а технічним хитрощам можна навчитися в процесі.

Спілкування з клієнтами

Перевага використання електронної пошти полягає у тому, що на отримані листи можна реагувати. Таким чином, ви маєте можливість отримати зворотні відгуки від своїх клієнтів та підписників. Це важливо для виправлення недоліків у розсилці та у роботі компанії. Основне завдання — активно відповідати на усі відгуки, щоб показати вашу готовність до постійного вдосконалення.

Звісно, email-маркетинг — це не лише листи, що продають. Електронні листи дають змогу нагадати про себе і сформувати стосунки, можуть вести клієнта лійкою продажів (зокрема, повністю автоматично). Вони використовуються для досліджень і опитувань, розсилки корисної інформації та багато чого іншого.

1.3 Плюси та мінуси email-маркетингу

Головні переваги email-маркетингу — це, звичайно, швидкість і адресність. За наявності якісно сегментованої бази клієнтів можна за дуже короткий час розіслати цільові листи найбільш зацікавленим сегментам.

До плюсів email-розсилок належать і широкі можливості персоналізації листів. Неважливо, скільки адресатів у розсилці — до кожного можна звернутися на ім'я, згадати контекст звернення і створити враження живого, індивідуального спілкування.

Ще одна перевага email-маркетингу — це гранично низька вартість розсилок. До слова, деякі сервіси дають можливість робити розсилки повністю безкоштовно. Так, NetHunt CRM дає змогу розсилати нелімітовану кількість листів через Gmail без додаткової оплати.

Автоматизованість — це той унікальний плюс розсилок, яким може похвалитися не так багато інструментів. Автоматизовані листи надсилатимуться клієнтам у потрібний момент самостійно. За допомогою email-розсилок можна створити і автоворонку продажів — тобто повністю автоматизований процес продажу, який відбувається без участі менеджерів.

Недоліки у розсилок теж є.

Істотний недолік у тому, що email-маркетинг можна використовувати, коли вже є напрацьована база контактів. Але і вона з часом застаріє і виснажиться. Перед тим, як зайнятися email-маркетингом, потрібно передбачити способи постійного поповнення бази клієнтів з інших джерел, оскільки самі розсилки не допоможуть наростити аудиторію.

Інші мінуси email-маркетингу умовні. Побоюючись розсилки спаму, чимало людей залишають компаніям електронну пошту, якою не користуються. У цьому разі на відкриваність листів ніяк не можна вплинути.

Крім цього, неграмотно проведені розсилки можуть призвести до потрапляння листів у спам.

Також варто врахувати, що багато поштових агентів розпізнають рекламні розсилки і поміщають їх в окрему вкладку або папку. Там листи ризикують загубитися серед багатьох інших.

Для ефективного email-маркетингу необхідні як мінімум два сервіси. По-перше, сервіс email-розсилок, з якого будуть надсилатися листи. Також може стати в пригоді конструктор листів, якщо базових дизайнів недостатньо.

1.4 Огляд існуючих систем керування розсилкою

sendpulse.ua

SendPulse надає можливість користувачеві використовувати спеціалізовані сервіси, включаючи сервіси для розсилок по Email, SMS, Viber, Web Push, WhatsApp, Facebook, Telegram, SMTP, валідації Email, а також створення та публікації сайтів (Лендінгів) і CRM — для управління продажами та взаємодії з клієнтами.

Користувач має можливість виконувати наступні дії та користуватися такими документами:

- ♣ Реєстрація облікового запису в обраному сервісі з можливістю зміни тарифного плану або видалення облікового запису за потреби.
- ♣ Право на отримання технічної підтримки.
- ♣ Доступ до навчальних матеріалів в Академії інтернет-маркетингу SendPulse.
- ♣ Можливість замовити додаткові послуги за умови додаткової оплати.
- ♣ Гарантія конфіденційності особистої інформації користувача та списків розсилок, завантажених в особистий обліковий запис клієнта.

- ♣ Доступ до партнерської програми сервісу-партнера RollerAds.

mailchimp.com

MailChimp — один із найпопулярніших сервісів поштових розсилок. Це англомовний сервіс і українського або російського перекладу в нього немає.

За допомогою сервісу можна:

- ❖ створювати новинні розсилки;
- ❖ автоматизувати надсилання листів;
- ❖ налаштувати інтеграцію контенту в соціальні мережі, наприклад, Facebook, Twitter;
- ❖ керувати базою контактів;
- ❖ аналізувати статистику email-кампаній;
- ❖ імпортувати дані в різні формати: Excel, Google Docs, .csv, .xml;
- ❖ налаштувати автоматичне оновлення даних, наприклад, під час роботи з файлами на Google Диску.

campaignmonitor.com

Налаштувати розсилку листів за допомогою Campaign Monitor просто, для цього не потрібні глибокі технічні знання. Впоратися можна навіть не маючи уявлення про те, що таке, наприклад, HTML. Усе налаштовується за допомогою шаблонів і перетягування потрібних блоків. Сервіс можна підключити до 250+ готових застосунків та інтеграцій: до CMS-движка, CRM або веб-сайту, наприклад, Shopify, Google Analytics, WordPress, WooCommerce, Salesforce, GetFeedback, Zendesk.

GetResponse — це польський сервіс email-розсилки. Платформа доступна на 26 мовах, але українського інтерфейсу поки що нема. Крім послуг розсилки та її автоматизації, компанія пропонує автоматичний генератор воронок продажу і лендінги. Є інтеграція з соціальними мережами і по API, галерея безкоштовних зображень, автовідповідачі листів та опитування, що вбудовуються в листи.

З першого погляду на ці найпопулярніші сервіси в даному напрямку здається що вже все є, навіщо вигадувати щось нове та витрачати час на реалізацію? Проте це лише перший погляд і витративши деякий час виникають думки «ось це не дуже зручно», «а де це знаходиться?», «за це я віддав гроші?». По перше всі ці сервіси працюють по підтвердженій базі контактів, тобто листи можна відправити лише тим, хто надає на це згоду. По-друге вони не орієнтовані для українського ринку, в кращому випадку є англійська. Вартість цих послуг починається від 150грн та за цю вартість ви отримуєте досить обмежені можливості щодо кількості відправлених листів чи розміру бази контактів.

2 ТЕОРЕТИЧНИЙ РОЗДІЛ

2.1 Вибір середовища програмування

#1 Visual Studio Code, розроблений Microsoft, - це легкий редактор вихідного коду, призначений для крос-платформної розробки веб- і хмарних додатків на Windows, Linux і macOS. Він має відладчик, інтеграцію з Git, підсвічування синтаксису, IntelliSense та інструменти для рефакторингу. Редактор пропонує широкі можливості кастомізації, такі як теми, прив'язки клавіш та конфігураційні файли. Хоча Visual Studio Code є безкоштовним програмним забезпеченням з відкритим вихідним кодом, готові дистрибутиви розповсюджуються під пропрієтарною ліцензією. Це потужний редактор, що підтримує JavaScript, TypeScript та Node.js, з широким спектром розширень для інших мов та середовищ виконання, таких як C++, C#, Java, Python, PHP, Go та .NET.

#2 Sublime Text - це власний текстовий редактор, який підтримує плагіни на мові програмування Python. Розробник дозволяє користувачам вивчати продукт безкоштовно, але для повноцінного використання потрібно придбати ліцензію. Sublime Text підтримує численні мови програмування і забезпечує підсвічування синтаксису для таких мов, як C, C++, C#, CSS, D, Dylan, Erlang, HTML, Groovy, Haskell, Java, JavaScript, LaTeX, Lisp, Lua, Markdown, MATLAB, OCaml, Perl, PHP, Python, R, Ruby, SQL, TCL і XML.

#3 JetBrains PhpStorm - це комерційне крос-платформне інтегроване середовище розробки для PHP, розроблене компанією JetBrains. PhpStorm - це інтелектуальний редактор для PHP, HTML і JavaScript, що включає в себе аналіз коду на льоту, запобігання помилкам у вихідному коді та інструменти автоматизованого рефакторингу для PHP і JavaScript. Редактор підтримує специфікації PHP 5.3/5.4/5.5/5.6/7.0/7.1, включаючи генератори, риси,

простори імен, закриття та синтаксис коротких масивів. PhpStorm базується на платформі IntelliJ IDEA, написаній на Java, і користувачі можуть розширювати його функціональність за допомогою плагінів, розроблених для цієї платформи, або створювати власні плагіни.

2.2 Середовище Visual Studio Code

Visual Studio Code — це редактор вихідного коду, який підтримує різноманітні мови програмування і надає розширені можливості, такі як підсвічування синтаксису, IntelliSense, рефакторинг, налагодження, навігація по коду та підтримка Git. Багато з цих можливостей можна налаштовувати за допомогою палітри команд або JSON-файлів, особливо при використанні настройок, які часто використовуються через цей інтерфейс. Палітра команд подібна до командного рядка і викликається через комбінації клавіш.

Visual Studio Code також дозволяє налаштовувати кодову сторінку при збереженні документа, а також символи перекладу рядка і мову програмування поточного документа. З початку 2018 року доступні розширення Python з відкритим вихідним кодом для Visual Studio Code, що надають розробникам розширені можливості для редагування, налагодження і тестування коду.

Розширення в Visual Studio Code представляють собою додаткові компоненти, які модифікують і покращують його функціональність. Вони можуть включати додаткові параметри, функції або сценарії для існуючих інструментів. З багатою кількістю розширень, доступних в Marketplace, розробники можуть вибирати ті, які відповідають їхнім потребам та підвищують продуктивність.

На березень 2019 року вбудований інтерфейс користувача дозволяє завантажувати та встановлювати тисячі розширень тільки в категорії "programming languages".

Таблиця 2.1 — Підтримка мов програмування

Можливості	Мови програмування
Підсвічування синтаксису	Batch file, C, C#, C++, CSS, Clojure, CoffeeScript, Diff, Dockerfile, F#, Git commit та rebase, Go, Groovy, HLSL, HTML, Handlebars, INI, JSON, Java, JavaScript, JavaScript React, LESS, Lua, Makefile, Markdown, Objective-C, Objective-C++, PHP, Perl, Perl 6, PowerShell
IntelliSense	CSS, HTML, JavaScript, JSON, Less, Sass, TypeScript
Рефакторинг	C#, TypeScript
Відладка	♣ JavaScript і TypeScript для проектів Node.js; ♣ C# і F# для проектів Mono (на Linux і MacOS); ♣ C і C ++; ♣ Python (при встановленому плагіні Python); ♣ PHP з XDebug (при встановленому плагіні PHP Debug); ♣ Java (при встановленому наборі плагінів Java Extension Pack або окремо встановленому Debugger for Java).

2.3 Вибір фреймворку для розробки клієнтської частини

1# Vue.js — це JavaScript-фреймворк з відкритим вихідним кодом, спрямований на створення користувацьких інтерфейсів. Він легко інтегрується в проекти з використанням інших JavaScript-бібліотек і може працювати як веб-фреймворк для розробки односторінкових додатків у реактивному стилі. Розробники вважають Vue.js прогресивним і більш поступово адаптованим порівняно з іншими веб-фреймворками. Його можна використовувати з базовими знаннями JavaScript і HTML, а також застосовувати Typescript. Vue.js пропонує шаблон MVVM та можливість прив'язки даних на JavaScript без потреби у ручному визначенні даних через

HTML-DOM. Цей фреймворк має власну досить багату документацію, доступну в кількох мовах, яка допомагає розробникам зрозуміти проєктування та розробку в браузері.

2# React - це JavaScript-бібліотека з відкритим вихідним кодом для розробки користувацьких інтерфейсів, розроблена Facebook, Instagram та спільнотою розробників. Вона дозволяє створювати односторінкові та мобільні додатки з високою швидкістю розробки, простотою і масштабованістю. React можна поєднувати з іншими бібліотеками, такими як MobX, Redux і GraphQL. Основними концепціями React є передача властивостей від батьківських до дочірніх компонентів, використання віртуального DOM для оптимізації оновлень інтерфейсу, а також використання JavaScript XML (JSX), що дозволяє використовувати HTML-подібний синтаксис для опису структури інтерфейсу. Часто React використовують у поєднанні з Redux для управління станами компонентів.

#3 Angular — це безкоштовний фреймворк для односторінкових веб-додатків на основі TypeScript з відкритим вихідним кодом, який розробляється командою Angular в Google та спільнотою приватних осіб і корпорацій. Angular повністю переписаний тією ж командою, яка створила AngularJS.

Однією з ключових особливостей Angular є те, що він використовує як мову програмування TypeScript. Тому перед початком роботи рекомендується ознайомитися з основами цієї мови, про які можна прочитати тут.

Але ми не обмежені мовою TypeScript. За бажання можемо писати додатки на Angular за допомогою таких мов як Dart або JavaScript. Однак TypeScript все таки є основною мовою для Angular.

2.4 React.js

React - це бібліотека для створення користувацьких інтерфейсів, і однією з її визначальних особливостей є використання JSX - мови програмування з синтаксисом, близьким до HTML, яка компілюється в JavaScript. Вона дозволяє розробникам досягати високої продуктивності за допомогою Virtual DOM. React надає можливість створювати ізоморфні додатки, що дозволяє уникнути ситуацій, коли користувач чекає завершення завантаження даних.

Однією з переваг React є можливість створення компонентів, які легко модифікувати і використовувати повторно в нових проектах. Це підвищує відсоток перевикористання коду, сприяє покриттю його тестами і підвищує рівень контролю якості. React Native дозволяє також створювати мобільні додатки для Android і iOS, використовуючи досвід розробки на JavaScript і React.

Ці технічні особливості можуть стати об'єктом роздумів для розробників. Тепер давайте перейдемо до наступного питання: Яку користь може отримати замовник від використання React?

Переваги використання React для замовника:

Підвищена продуктивність високонавантажених додатків: Використання Virtual DOM може підвищити продуктивність додатків з великою кількістю даних, знижуючи ймовірність незручностей і покращуючи користувацький досвід.

Швидший рендеринг сторінок за рахунок ізоморфного підходу: Ізоморфний підхід дозволяє рендерити сторінки швидше, поліпшуючи враження користувачів та полегшуючи індексацію сторінок пошуковими

системами. Крім того, цей підхід дозволяє уникнути дублювання функціоналу і знизити час розробки.

Перевикористання коду для мобільних додатків: Код, написаний для веб-сайту, може бути повторно використаний для створення мобільного додатку, що полегшує розробку і уникнення найму окремих команд розробників для веб та мобільних платформ.

Коли ми говоримо про ізоморфні застосунки або ізоморфний JavaScript, маємо на увазі використання одного коду як у серверній, так і в клієнтській частині застосунку. Це полегшує швидкість завантаження сторінок, покращує індексацію пошуковими системами та знижує час розробки.

Ці переваги роблять React привабливим вибором для замовників, які прагнуть поліпшити користувацький досвід своїх веб-сайтів та додатків, прискорити розробку та використовувати сучасні технології розробки.

Document Object Model (DOM) - це спосіб представлення і взаємодії з об'єктами в HTML, XHTML і XML документах. Згідно з цією моделлю, кожен документ є ієрархічним деревом елементів, відомим як DOM-дерево. Ми можемо отримати доступ і змінювати елементи документа, використовуючи спеціальні методи. При створенні динамічних інтерактивних веб-сторінок ми хочемо, щоб DOM оновлювався швидко після змін стану елементів. Деякі фреймворки використовують "dirty checking" - регулярне опитування стану документа та перевірку змін у структурі даних. Проте це може стати проблемою для високонавантажених додатків.

Virtual DOM вирішує цю проблему, зберігаючи віртуальну копію DOM в пам'яті. Коли "справжній" DOM змінюється, React оновлює Virtual DOM швидко. React порівнює зміни зі станом DOM і тільки після цього відбувається перемальовування змінених компонентів.

Цей підхід дозволяє уникнути регулярного оновлення DOM, що призводить до вищої продуктивності React-додатків, особливо у

високонавантажених сценаріях. Крім того, через ізоморфну природу React, можливий рендеринг як на стороні сервера, так і на стороні клієнта.

Перевикористання коду також сприяє більш ефективному розробленню і тестуванню застосунків. Здійснюючи рендеринг на обох сторонах, React розробники можуть ефективно використовувати той самий код для створення та тестування компонентів, зменшуючи зусилля і час, необхідний для розробки та підтримки додатків.Мобільні додатки мають деякі переваги порівняно з сайтами. Їх можна використовувати без з'єднання з Інтернетом. Вони мають доступ до таких можливостей пристрою, як спливаючі сповіщення. Також вони дають змогу бути в контакті з вашими користувачами в режимі 24/7. React Native — це фреймворк, який дає змогу вам створювати мобільні додатки, використовуючи React. Логіка додатка пишеться на JavaScript, таким чином, програмісту не потрібно відмовлятися від звичних прийомів веб-розробника. Усе що потрібно — навчитися писати специфічний для пристрою код, який адаптує компоненти, раніше створені для вебсайту, до нового середовища проживання.

Якщо ми порівняємо витрати на розробку різних видів мобільних додатків, ми отримаємо приблизно такі результати:

У випадку з нативними додатками ви можете сподіватися на досить високу продуктивність, але вартість розроблення буде досить високою;

Якщо ви віддасте перевагу фреймворкам, які дають змогу використовувати HTML5, CSS3 і JavaScript, наприклад PhoneGap, ви можете знизити вартість. Але в цьому разі рівень продуктивності буде набагато нижчим;

У разі React ви можете досягти рівня продуктивності, порівнянного з нативними додатками. При цьому вартість розробки можна порівняти з попереднім прикладом.

Якщо ви розглядаєте можливість створення корпоративного веб-додатка і сумніваєтеся, чи варто розробляти його мобільну версію, React

Native може бути відмінним варіантом. Цей фреймворк дозволяє використовувати наявну логіку веб-додатка для створення мобільного додатка, спільно використовуючи той самий код, який використовувався при створенні веб-сайту.

Переваги використання React Native для мобільних додатків у контексті корпоративних веб-проєктів:

Ефективне перевикористання коду: Використовуючи той самий код для веб-додатка і мобільного додатка, ви прискорюєте процес розробки та уникаєте дублювання коду. Це забезпечує більш швидкий розвиток мобільної версії та зменшує ймовірність помилок.

Менше помилок при перевикористанні компонентів: Якщо ви маєте добре спроектовані та перевірені компоненти, які використовуєте повторно, вам потрібно буде писати менше нового коду при створенні нового інтерфейсу. Це не лише зменшує кількість потенційних помилок, але й полегшує розуміння та виправлення помилок, якщо вони все ж виникають.

Такий підхід сприяє не лише ефективній розробці, але й полегшує підтримку та оновлення вашого корпоративного додатка на різних платформах.

2.5 Вибір мови для розробки серверної частини

#1 Node.js — це однопотокове кросплатформенне середовище виконання з відкритим вихідним кодом і бібліотека, що використовується для запуску веб-додатків, написаних на JavaScript, поза браузером клієнта. Звучить трохи складно, правда? Простіше кажучи, Node.js — це програмне середовище, яке дає змогу запускати програми, написані мовою Javascript, поза браузером. Історично програми, написані Javascript, на відміну від інших мов програмування, можна було запустити тільки в браузерах, які мали спеціальний вбудований рушій виконання коду цієї мови. Поза браузером

Javascript, можна сказати, не працював. Під час розробки Node.js за основу було взято рушій виконання JavaScript під назвою V8, який був створений компанією Google і використовувався в браузері Google Chrome. Оскільки після створення Node.js Javascript код можна запустити фактично в будь-якому середовищі, за допомогою цієї бібліотеки можна написати не тільки фронтенд, а й серверну частину веб-додатка. Простіше кажучи, це означає, що цілі сайти тепер можуть працювати з використанням єдиного "стека", що робить розробку та обслуговування набагато швидшим і легшим, даючи змогу зосередитися на досягненні бізнес-цілей проекту. Node.js має відкритий вихідний код, тому працювати з ним можна абсолютно безкоштовно. Його й сьогодні продовжує розвивати та покращувати глобальна спільнота розробників. Важливо розуміти, що Node.js насправді не фреймворк і не бібліотека, як у випадку з традиційним програмним забезпеченням, а середовище виконання. Він є легким, гнучким і простим у розгортанні, а всі його функції допоможуть оптимізувати та прискорити ваш додаток.

#2 Мову C# розробила компанія Microsoft і вперше вона з'явилася ще 2000 року. З моменту її появи минуло вже багато років. Мова перетерпіла велику кількість оновлень і нововведень. На сьогоднішній день мова є однією з найпопулярніших і затребуваних мов у світі.

Мова C# є об'єктно орієнтованою мовою програмування. Це означає, що кожен файл являє собою певний клас.

Сама мова використовує синтаксис, що сильно нагадує мову Сі або ж Java. Так насправді вийшло не спроста. Мова C# хоч і є досить старою мовою, але порівняно з мовами C++ і Сі є дитиною.

Мова перейняла багато чого від своїх попередників — мов C++, Delphi, Smalltalk і, особливо, Java. Під час розробки C# було взято найкращі моменти з усіх цих мов. Наприклад, C# на відміну від C++ не підтримує множинне успадкування класів. Так було вирішено через їхню незручність використання.

#3 PHP — С-подібна скриптова мова загального призначення, яку інтенсивно застосовують для розробки веб-додатків. Нині підтримується переважною більшістю хостинг-провайдерів і є однією з лідерів серед мов, що застосовуються для створення динамічних веб-сайтів. PHP є мовою програмування з динамічною типізацією, яка не потребує зазначення типу під час оголошення змінних, так само як і самого оголошення змінних.

Ключове слово class було зарезервовано ще в третій версії мови. У четвертій версії стало можливо створювати класи та об'єкти на їхній основі. Однак принципи ООП підтримувалися лише частково, так, наприклад, усі члени (змінні та методи) були відкриті. До того ж створення об'єктів було дорогою операцією і працювало повільно.

Починаючи з п'ятої версії PHP має повну підтримку ООП. Робота з класами була оптимізована і тепер такий код працює досить швидко.

2.6 PHP

PHP (рекурсивний акронім словосполучення PHP: Hypertext Preprocessor) — це популярна відкрита мова програмування загального призначення, спеціально призначена для веб-розробок. Код PHP може бути вбудований безпосередньо в HTML, спрощуючи процес створення веб-сторінок.

Наприклад, замість виведення HTML-коду командами мови, як у Perl або C, PHP використовує скрипти, що містять HTML з вбудованим PHP-кодом. Код PHP виділяється спеціальними тегами <?php і ?>, які визначають початок і кінець PHP-режиму.

Одна з особливостей PHP полягає в тому, що він виконується на сервері та генерує HTML, який надсилається клієнту. Це відрізняє PHP від JavaScript, який виконується на браузері клієнта. Клієнт отримує лише результат виконання PHP-скрипта, не відомий конкретний код, який його створив.

PHP вважається простим для вивчення, але при цьому він забезпечує достатню гнучкість для задоволення потреб і початківців, і професіоналів у програмуванні. Легко розпочати роботу з PHP, і за короткий час вже можна створювати прості PHP-скрипти.

Хоча PHP в основному призначений для веб-розробок, його можливості застосування не обмежуються лише цією сферою.

3 СИНТЕЗ СИСТЕМИ РОЗСИЛКИ ЕЛЕКТРОННИХ ЛИСТІВ

3.1 Формулювання технічних вимог до системи керування розсилкою електронних листів

Для розробки програмного забезпечення з розсилки електронних листів важливо враховувати ряд технічних вимог для забезпечення ефективності та безпеки системи.

По-перше, програма повинна мати інтуїтивно зрозумілий інтерфейс для користувачів, що дозволить легко налаштовувати та відстежувати розсилку. Наявність інструментів для імпорту та управління списками адресатів також є обов'язковою вимогою.

Важливо забезпечити можливість персоналізації листів, включаючи вставку індивідуальних даних, а також підтримку HTML-формату для створення креативного вмісту листів. Система повинна бути сумісна з різними поштовими сервісами та гарантувати доставку листів відповідно до стандартів електронної пошти.

Забезпечення безпеки даних є пріоритетом, тому програма повинна використовувати шифрування для захисту інформації про адресатів та відправників.

Щоб забезпечити оптимальну швидкодію системи, програма повинна оптимізувати процес розсилки, використовуючи ефективні алгоритми та механізми кешування. Технічною вимогою також є можливість налаштовування параметрів доставки, таких як інтервали розсилки.

Усі технічні вимоги повинні відповідати стандартам безпеки та конфіденційності, забезпечуючи високий рівень захисту особистих даних користувачів та конфіденційної інформації.

3.2 Формулювання технічних вимог до обладнання

3.2.1 Вимоги до серверного обладнання

Ємність сховища даних. Сервер повинен мати достатньо обсягу сховища даних для зберігання великої кількості вхідних даних. З урахуванням розміру проекту рекомендується мінімально 500 МБ вільного простору.

Потужність процесора. Для забезпечення швидкодії виконання PHP-сценаріїв та оптимальної роботи React-компонентів, сервер повинен мати потужний процесор з високою частотою тактування, що забезпечить швидку обробку запитів.

Оперативна пам'ять (RAM). Рекомендується мінімум 4ГБ оперативної пам'яті для ефективної обробки запитів та запобігання збоїв в роботі програми через нестачу пам'яті.

3.2.2 Вимоги до мережевого обладнання

Швидкість передачі даних. Мережеве обладнання повинно мати достатню пропускну здатність для ефективного обміну даними між клієнтом (React) та сервером (PHP). Рекомендована швидкість передачі даних - не менше 100Мбіт/с. Цього можна досягти розмістивши клієнтську та серверну частину на одному сервері.

Стабільність мережі. Для забезпечення стабільності роботи додатку важливо мати надійне і стабільне мережеве з'єднання без перебоїв та втрат пакетів даних.

3.2.3 Вимоги до серверного програмного забезпечення

Підтримка PHP 8.1. Сервер повинен мати встановлену та налаштовану версію PHP 8.1 для коректної роботи серверної частини додатку.

Підтримка віртуалізації. Якщо використовується віртуалізація, серверне обладнання повинно підтримувати цю технологію для ефективного розгортання та масштабування додатку.

3.3 Вибір та обґрунтування застосування апаратних засобів

З урахуванням технічних вимог до серверного обладнання та програмного забезпечення, Linux веб-хостинг – найкращий вибір. Операційна система Linux володіє стабільністю та безпекою, що робить її оптимальним середовищем для ефективної роботи програмного забезпечення з розсилки електронних листів. Використання Linux також забезпечить високий рівень сумісності з відповідними версіями PHP та іншими необхідними компонентами для забезпечення надійності та швидкодії системи.

На Linux веб-хостингах зазвичай передвстановлена адміністративна панель, наприклад, cPanel, що значно полегшує налаштування серверу.

3.4 Синтез схеми системи

3.4.1 Синтез програмної схеми системи

Загальна архітектура системи визначається послідовністю подій та функціональністю, що забезпечують користувачам зручну та ефективну взаємодію з системою керування автоматизованою розсилкою повідомлень. Описана структура включає в себе наступні етапи:

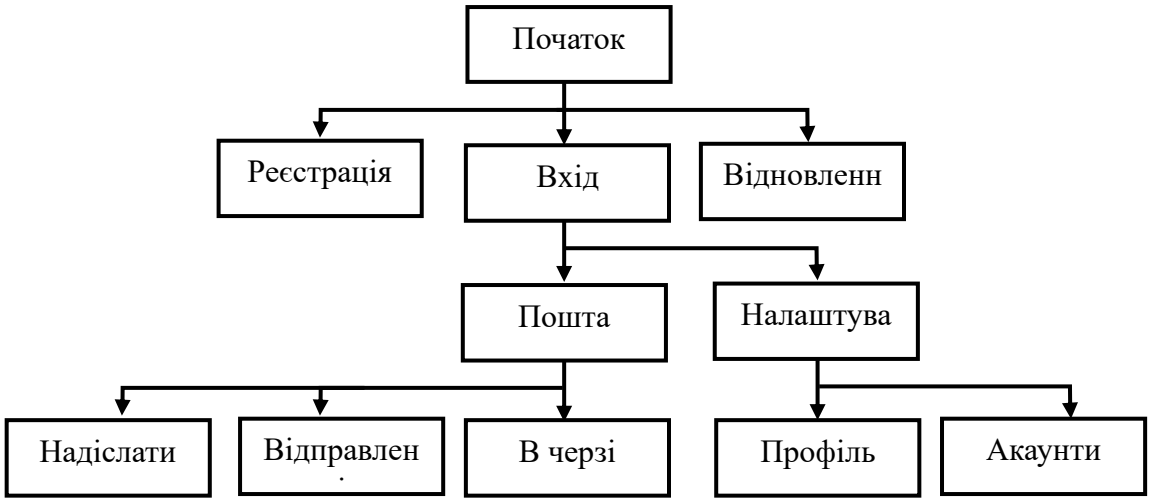
Вхід на сайт. Користувач потрапляє на сайт, де розміщена система керування розсилкою повідомлень. Цей етап включає в себе введення URL сайту в браузер та перехід на головну сторінку.

Опції для користувача. Користувач має три основні опції: реєстрація, вхід або відновлення паролю. Якщо у користувача вже є акаунт, він може використовувати свої дані для входу.

Головне меню системи. Після входу користувач потрапляє до головного меню системи. Тут він має можливість обирати подальші дії, такі як відправлення повідомлень, перегляд вже надісланих або очікуючих відправок, або перехід до розділу налаштувань.

Взаємодія з головним меню. Користувач може залишитися на головному меню, де може здійснювати різноманітні функції. Основні можливості включають відправлення повідомлень та перегляд їх статусів.

Розділ налаштувань. Користувач може обрати перехід до розділу налаштувань, де доступні дві основні категорії. Перша - "Налаштування профілю", де можна змінити пароль та налаштувати шаблон листа. Друга - "Налаштування", де користувач може додавати чи видаляти акаунти для



розсилки повідомлень.

Рисунок 3.1 – Структурна схема клієнтської частини системи

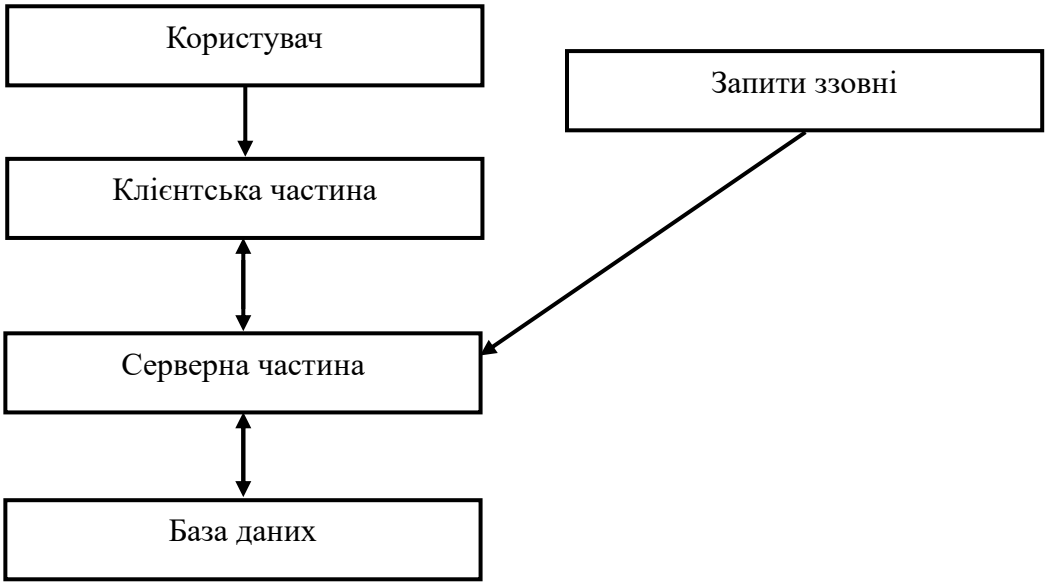


Рисунок 3.2 – Функціональна схема програмної частини системи

3.4.2 Синтез апаратної схеми системи

Синтез апаратної схеми системи контролю автоматизованої масованої розсилки електронних листів визначається необхідністю ефективного обміну інформацією між різними пристроями, такими як персональні комп'ютери (ПК), сервери та мережеве обладнання. Правильний вибір апаратної складової та оптимальна схема розміщення є ключовими факторами для забезпечення швидкодії та надійності всієї системи. Вимоги до обладнання:

Серверна архітектура. Система базується на централізованому сервері, який відповідає за обробку та збереження великого обсягу даних, таких як дані користувачі, дані отримувачів, текст повідомлень тощо.

Використовуються високопродуктивні сервери з достатньою кількістю ядер, оперативної пам'яті та накопичувального простору для оптимальної обробки та зберігання даних.

Мережеве обладнання. Забезпечується стійке та швидке підключення пристроїв до сервера для передачі даних. Використовуються високошвидкісні комутатори та маршрутизатори для оптимізації передачі даних в мережі.

Моніторинг та управління. Сервер повинен мати вбудовані засоби моніторингу для відстеження стану сервера та мережі.

Загальна апаратна схема повинна забезпечувати високу продуктивність, надійність та зручність в управлінні системою автоматизованої масованої розсилки електронних листів.

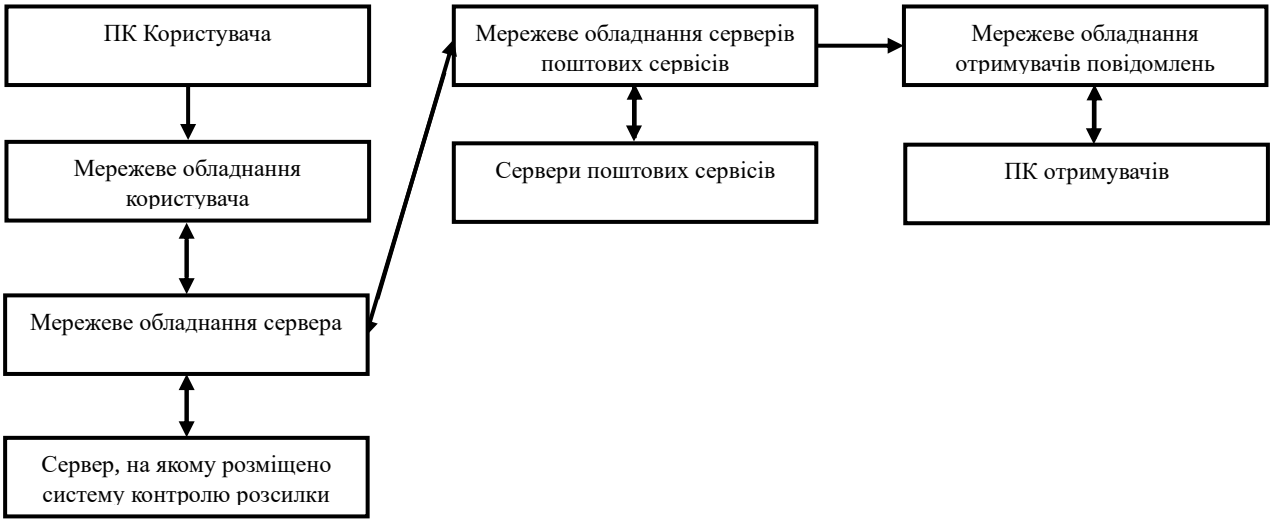


Рисунок 3.3 – Функціональна схема апаратної частини системи

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

4.1 Призначення й область застосування програмного забезпечення

4.1.1 Призначення

Це програмне забезпечення призначене для автоматизації та полегшення процесу масової розсилки електронних листів у веб-додатку. Основна функціональність дозволяє користувачам завантажувати базу контактів у форматі .csv, використовуючи конструктор, налаштовувати зміст листів та планувати час відправлення. Застосування додатку включає, але не обмежується:

- ♣ Маркетинг та Реклама: Ефективна реалізація кампаній електронної пошти для просування товарів та послуг.
- ♣ Інформування клієнтів: Відправлення оновлень, спеціальних пропозицій та інших повідомлень клієнтам.
- ♣ Зв'язок зі співробітниками: Можливість розсилки внутрішніх повідомлень та оголошень усім співробітникам.

4.1.2 Область застосування

Це програмне забезпечення підходить для широкого спектру сфер, включаючи бізнес, маркетинг, освіту та громадську діяльність. Відправка персоналізованих листів у великій кількості дозволяє здійснювати ефективний інтерактивний контакт з аудиторією без значних зусиль.

4.2 Обгрунтування технічних характеристик програм

4.2.1. Visual Studio Code (версія 1.84.2)

Системні Ресурси: Visual Studio Code є легким та ефективним інструментом, має низькі вимоги до оперативної пам'яті та процесора. Рекомендується пристрій із ОЗУ не менше 1.5 ГБ, вільним простором на системному диску не менше 1 ГБ та роздільною здатністю екрану не нижче 1024*768 пікселів.

Операційна Система: Підтримується на основних операційних системах, таких як Windows, macOS, та Linux, забезпечуючи гнучкість у виборі платформи.

4.2.2. Node.js (версія 18.15.0)

Операційна Система: Підтримується на основних операційних системах, що дозволяє розробникам використовувати його на різних платформах без обмежень.

Системні Ресурси: Рекомендовано мати не менше 512 МБ оперативної пам'яті, але більше RAM полегшить роботу сервера та ваших додатків. На системному диску повинно бути не менше 200 МБ вільного місця.

4.2.3. Xampp (версія 8.1)

Операційна Система: Підтримується на основних операційних системах, що дозволяє розробникам використовувати його на різних платформах без обмежень.

Системні Ресурси: Рекомендовано мати не менше 512 МБ оперативної пам'яті, але більше RAM полегшить роботу сервера та ваших додатків. На системному диску повинно бути не менше 2 ГБ вільного місця.

4.3 Опис розробленої програми

4.3.1 Розробка структурної схеми

Загальна структура системи (Рисунок 4.1) виглядає наступним чином: користувач потрапляє на сайт, де розміщується система керування автоматизованою розсилкою повідомлень, після чого є три варіанти подальших дій: реєстрація, вхід та відновлення паролю. Якщо користувач вже має акаунт і пам’ятає дані від цього акаунту, то входить і потрапляє до головного меню системи. Далі, на вибір, користувач може залишитися на головному меню, звідки може відправити повідомлення, переглянути надіслані, чи ті що ще очікують відправки, чи може перейти до іншого розділу – налаштування. У налаштуваннях також є розподілення: налаштування профілю (зміна пароля та налаштування шаблона листа) та налаштування, тобто додавання чи видалення акаунтів, з яких можна робити розсилку повідомлень.

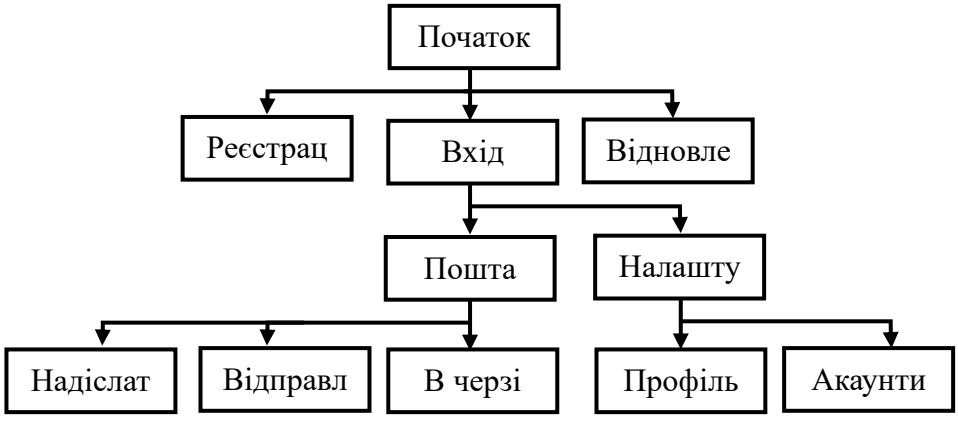


Рисунок 4.1 – Структурна схема програми

Структура бази даних (Рисунок 4.2) складається з п’яти таблиць:

- users – таблиця користувачів програмного забезпечення;
- messages – таблиця надісланих повідомлень;
- accounts – таблиця акаунтів, що використовуються для розсилки;
- tasks – таблиця повідомлень, що очікують відправки;
- contacts – таблиця з контактами для розсилки.

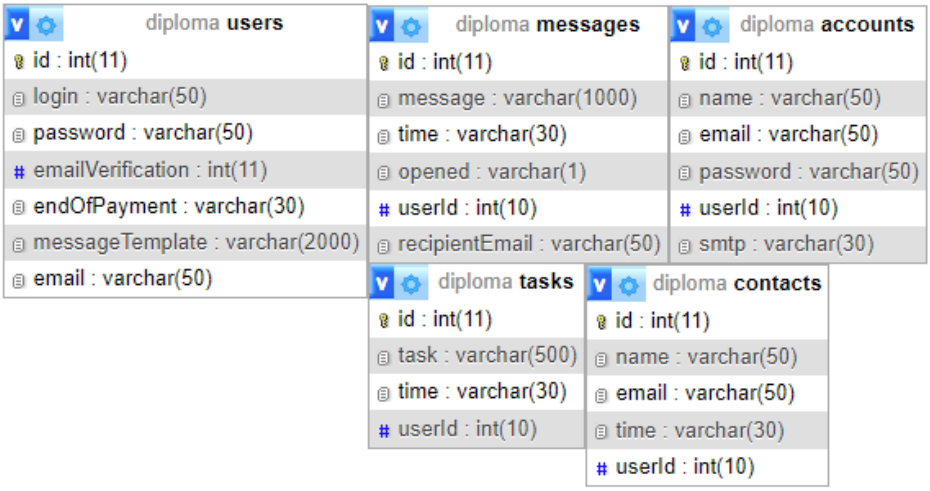


Рисунок 4.2 – Структура бази даних

4.3.2 Розробка дизайну

Перед початком розробки клієнтської частини було розроблено макет програми. Це достатньо важкий етап через те, що дизайн повинен бути оригінальним та виглядати однаково гарно на різних пристроях.

Було переглянуто низку ресурсів, що надають послуги по відправці одиночних або масованих електронних листів, щоб створити не просто гарний дизайн, але і з зручним інтерфейсом, що прописаний багатолітнім користувацьким досвідом.

Останнім часом все більше і більше додатків починають підтримку темних тем користувацького інтерфейсу, тому було вирішено взяти відразу темну стилістику інтерфейсу з м’якою кольоровою гамою, яка не перенапружує зір в будь-який час доби. Розроблений наступний макет для зручності подальшої розробки інтерфейсу клієнтської частини:

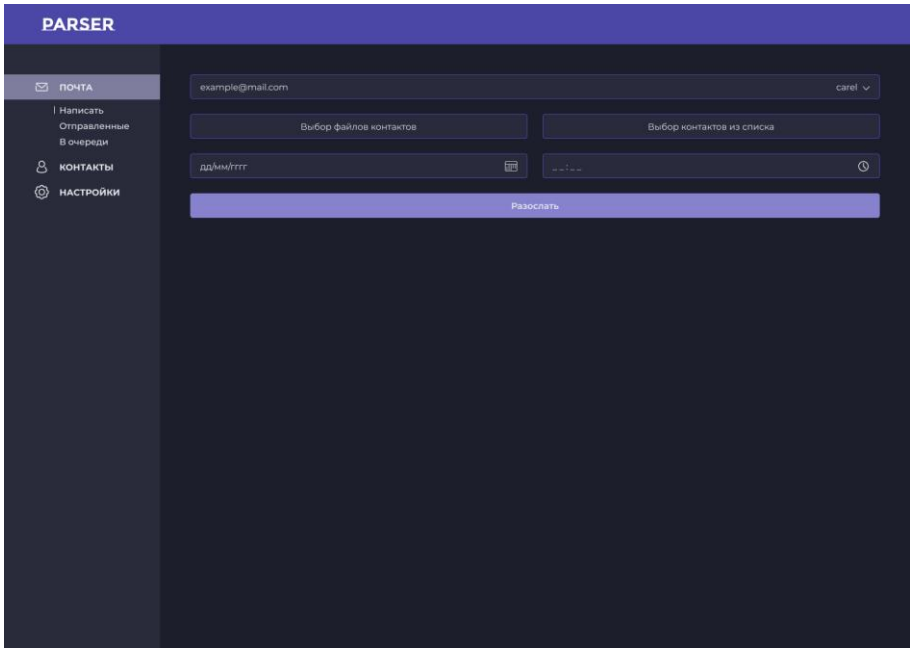


Рисунок 4.3 – Макет користувацького інтерфейсу

4.3.3 Розробка клієнтської частини

Клієнтська частина – це частина програмного продукту, з якою взаємодіє користувач. В даному проєкті було реалізовано з боку клієнтської частини макет, котрий реалізовано статичного розміру, щоб не було зайвих рухів і додаток був максимально схожий на звичайну desktop програму, а не веб-сайт. Статичність розміру було досягнуто за допомогою css властивості «display: grid» та заданих розмірів, що відповідають висоті та ширині дисплею.

Розробка клієнтської частини завжди починається з генерації початкової структури проєкту, в React.js це можна зробити за допомогою команди “npx create-react-app .” в терміналі. Команда виконається успішно,

якщо встановлений node.js та назва директорії, в якій відбувається створення проєкту, написана нижнім регістром латиницею.

Після успішної генерації «скелету» проєкта в папці src створюємо додаткові директорії для зручного відображення структури:

- components – для зберігання файлів компонентів;
- contexts – для зберігання контекстів (глобальних змінних, що доступні з будь-якого місця проєкту);
- services – для зберігання функцій, що можуть буди використані в різних компонентах.

Наступними кроками йде верстання користувацького інтерфейсу згідно з розробленим макетом, після чого – розробка функцій для більш глибокого налаштування інтерфейсу, таких як: пошук, пагінація даних тощо, – а також функціонал для «спілкування» клієнтської частини програмного забезпечення з серверною частиною.

При потраплянні користувача до системи, в першу чергу відбувається перевірка токена: якщо токен існує, то перевіряється чи він дійсний (відправляється запит на сервер для перевірки токена) і у разі якщо він дійсний, то користувача перенаправляється до інтерфейсу системи, якщо токен недійсний або відсутній, то нічого не змінюється, система очікує на дії користувача. Це реалізовано у функції src/service/enter/checkToken.js.

При вході в систему є 3 варіанти дій: вхід в акаунт, реєстрація акаунту, відновлення паролю. Всі 3 варіанти обробляються в файлі src/service/enter/enter.js. Спочатку відбувається валідація даних, що були введені, якщо дані коректні, то формується тіло запиту, що складається з GET та POST параметрів, після чого відправляється запит на сервер для обробки даних. В залежності від відповіді сервера виводяться сповіщення про помилки, або відбувається вхід в акаунт, якщо дані вірні.

Спливаючі сповіщення (Рисунок 4.4). В проєкті використовується глобальна змінна toastContext, в ній зберігаються масиви об’єктів даних.

Об’єкт складається з полів: type, text. Type – це тип сповіщення. Є три типи: успішний (success) зеленого кольору, в процесі (inProcess) жовтий, помилка (error) червоного кольору. Незалежно від місцезнаходження користувача на сайті, відображається стек сповіщень в правому нижньому куту, одночасно до 5 штук. Додаються вони динамічно за рахунок jsx властивостей, тобто проект переглядає зміни в toastContext, якщо він змінився, то на сайті відразу відображаються зміни. Закриття сповіщень реалізовано видаленням об’єкту з масиву.

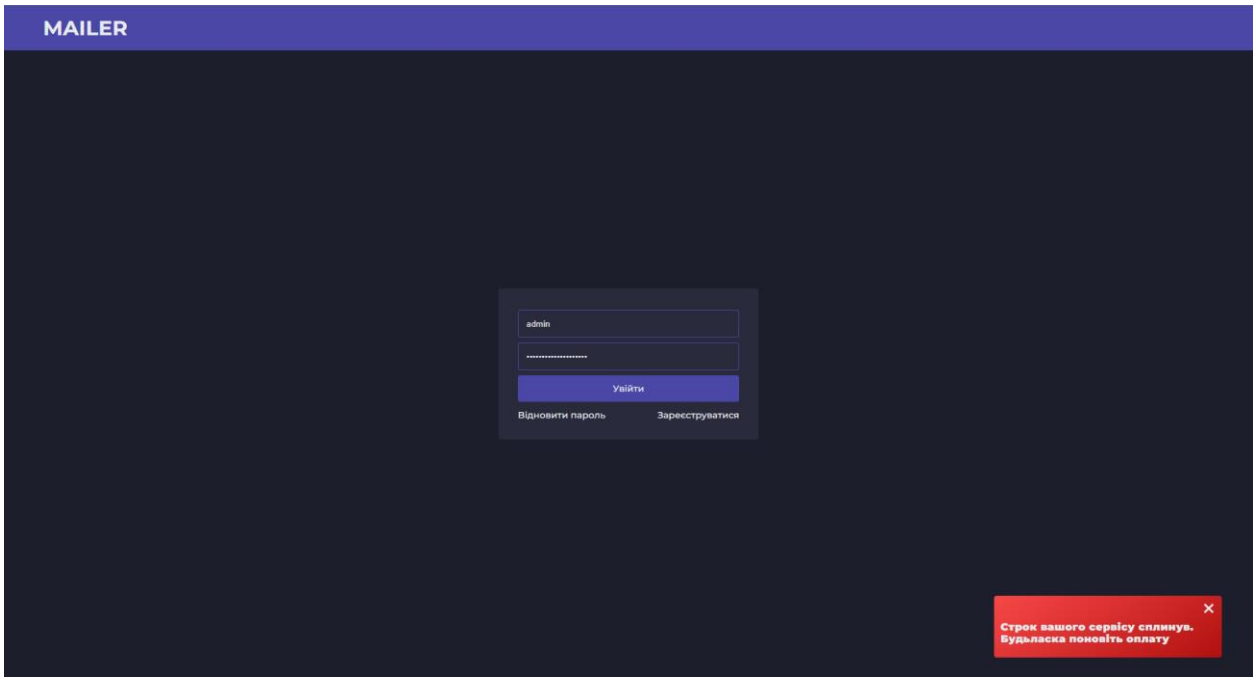


Рисунок 4.4 - Сповіщення про завершення оплати сервісу

Пагінація розроблена окремим компонентом «Paginator», для можливості перевикористання. В компонент передаються кількість сторінок, поточний номер сторінки та функція, що викликається при натискання кнопки гортання сторінки. Пагінація розрахована на 10 елементів та працює наступним чином: виводяться дані в таблицю по формулі *перший_елемент = номер сторінки * 10*, *останній_елемент = номер сторінки * 10 + 10* . У разі, якщо даних недостатньо для гортання, кнопки неактивні.

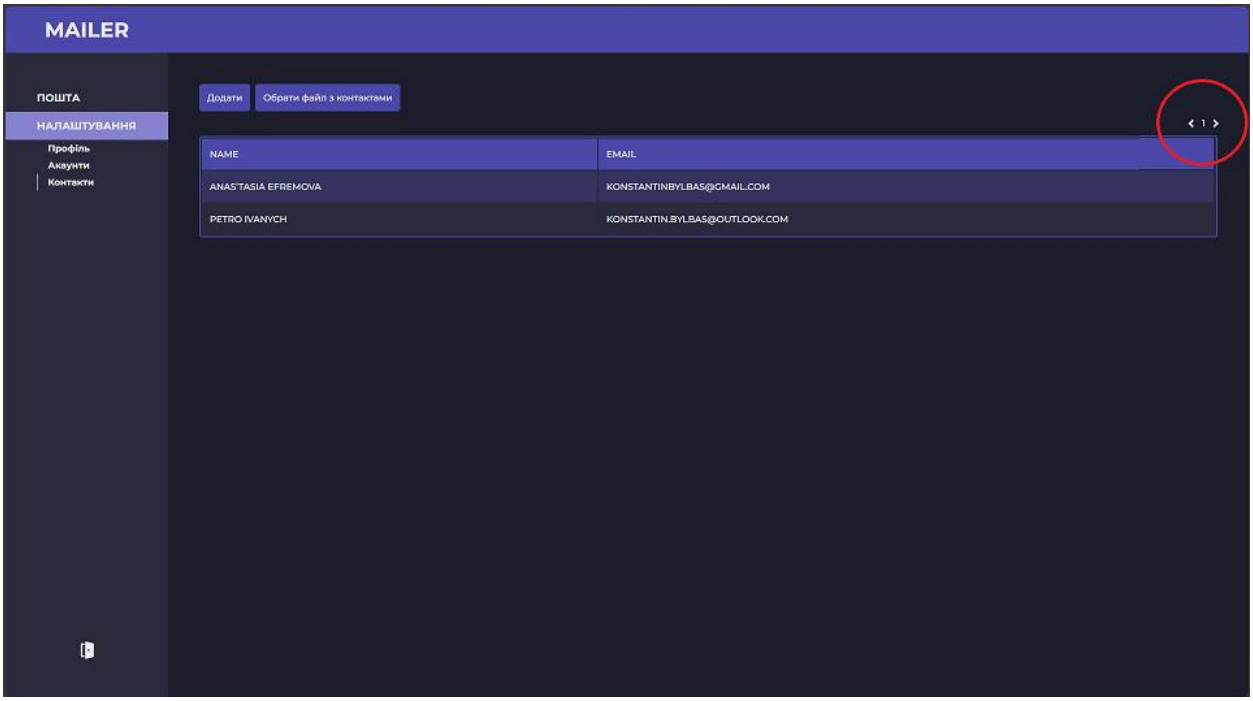
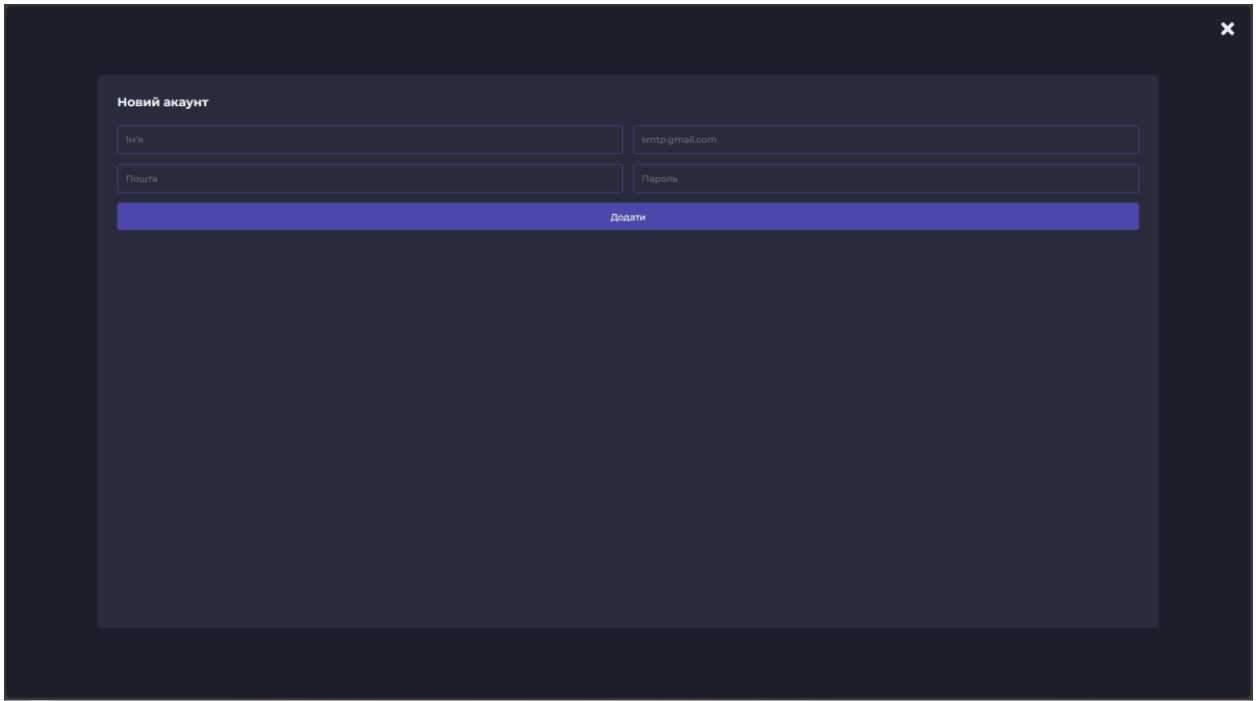


Рисунок 4.5 – Пагінація даних

Модальні вікна реалізовані схожим принципом, що й спливаючі



сповіщення – використовується змінна modalContext.jsx для можливості виклику модального вікна з будь-якого місця. Замість масиву з повідомленнями, тут використовується один об’єкт, що зберігає в собі налаштування, такі як: show, type, tab, context, setContext. Через те, що контекст модального вікна зберігає лише один об’єкт даних, одночасно можна викликати лише одне модальне вікно. Модальні вікна використовується для різних завдань: додавання контактів, додавання акаунтів, обирання контактів для розсилки електронних листів.

Рисунок 4.6 – Модальне вікно

Додавання контактів з файлу. Коли користувач обирає файл, в полі вибору файла, яке приховане стилями і викликається кнопкою, спрацьовує onChange() подія, дані відправляються на обробку в src/services/emailSending/ReadFile.js. Спочатку функція знаходить потрібні колонки, якщо у файлі є зайві дані, після чого проходить по усім рядкам та зчитує потрібні дані (електронну пошту та ім’я) в масив, який згодом відправляється на сервер для додавання в таблицю контактів.

Відправка повідомлень знаходиться на головній сторінці користувача, тому що це найважливіша і більш використовувана сторінка. Для відправки повідомлення треба додати хоча б один акаунт, з якого буде відбуватися відправка листів, додати шаблон повідомлення який повинен бути довший за 10 символів та коротший ніж 1000 символів. Після цих кроків треба обрати контакт(и), яким буде надіслано повідомлення, згідно шаблону обрати час та дату, якщо треба відправити з затримкою, чи залишити ці поля без змін, якщо відправка потрібна в цю ж мить. Дані обробляються в файлі src/services/emailSending/PreparationForSend.js. Спочатку відбувається збір необхідних даних (акаунт, контакти, дата та час) та формується масив даних, який відправляється на сервер. Дані відправляються на сервер по-елементно паралельно через те, що PHP асинхронна мова, обробка навіть 50 контактів

займає декілька хвилин, а якщо відправити ті ж листи паралельно з клієнта, то час на обробку становить 10 секунд. Якщо клієнтом було обрано дату та час відправки і вказаний момент часу більший за поточний, то дані відправляються з параметром додавання у чергу, а не на відправку.

Налаштування. Додавання, редагування, видалення даних у підвкладках вкладки налаштування відбувається шляхом валідації даних та запитом на сервер з відповідними параметрами та даними.

Пошук контактів у списку відправлених реалізований таким чином: перебирається кожен рядок (кожен контакт) та перевіряється значення в інпуті, що призначений тільки для читання, якщо значення у рядку пошуку співпадає хоча б частиною значення в одному з полів введення даних, то такий рядок залишається у списку, інші приховуються додаванням класу «dNone» до рядку.

Після завершення розробки клієнтської частини компілюємо (збираємо готовий варіант програми) за допомогою команди «npm run build», яка активує скрипт, прописаний в файлі package.json, за замовчування виконується «react-scripts build». Готовий варіант програми складається до папки build і для його успішного запуску на сервері – достатньо розмістити файли з директорії build в корінь папки, до якої йде звернення при звертанні до сервера.

4.3.4 Розробка серверної частини

Серверна частина працює за принципом: запит – результат, тобто з клієнтської частини надходить запит з параметрами, а сервер опрацьовує дані і завжди надає результат. В головному файлі зберігаються підключення більшості функцій та ендпоінти, або маршрути, тобто саме ті точки, куди звертається клієнт, а також параметри цього файлу, щоб він відпрацьовув як API. Ендпоінти працюють за принципом: отримав тип звернення через GET параметри та направив дані далі в потрібну функцію/клас. Маршрути розділені за допомогою конструкції switch для кращої читабельності коду. На початку файлу прописані хедери для того, щоб цей файл міг приймати запити і віддавати відповідь, як API.

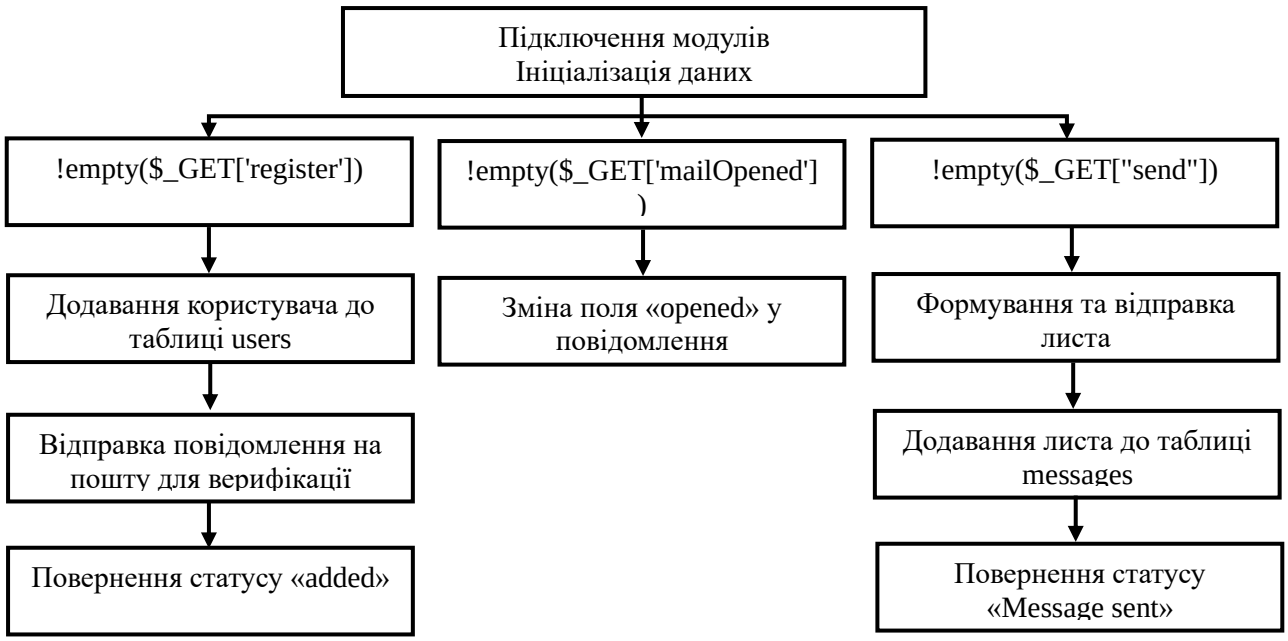


Рисунок 4.7 – Структурна схема роботи серверної частини системи

Підключення до БД відбувається в класі Database. Винесені окремо змінні для того, щоб вказати хост, назву бази даних, ім’я користувача бази даних та пароль користувача БД. При зверненні до функції getConnection, виконується запит до БД за допомогою вбудованого модуля PDO(), який використовується при кожному зверненні до БД.

Обробка даних, що зберігаються в базі даних відбувається у класі Data з набором різних методів для обробки цих даних: getData(), deleteData(). changeData() тощо. Get, delete, та change лише виконують відповідні запити до БД. Метод addData() формує рядок стовпців для кожної таблиці БД, виконує перевірку на наявність дублікату даних для таблиць users та contacts, для таблиць contacts та tasks додає в масив значень значення поточної дати та часу, якщо таблиця tasks, то змінює контакти, які додаються в чергу для відображення на боці клієнта у вкладці «В черзі», далі масив даних переписується у рядок для можливості створення запиту до БД та відправляється запит, якщо запит до БД був коректний і таблиця users, то відправляється лист користувачу на пошту для подальшого підтвердження пошти і в кінці повертається рядок «added», щоб клієнтська частина розуміла, що запит успішно відпрацював. Метод restorePassword() перевіряє наявність користувача в БД, генерує новий пароль, змінює пароль у БД та відправляє користувачу на пошту новий пароль від акаунта.

Перевірка користувача відбувається в окремому класі – Login. Тут реалізовані методи для перевірки наявності користувача, перевірки токена, отримання токена. Перевірка користувача реалізована шляхом отримання всіх користувачів, перебором кожного користувача та порівняння логіну з паролем з кожним записом. Перевірка токена зроблена шляхом створення нового токена на основі даних, що надійшли і порівняння з поточним токеном. Формування токена відбувається шляхом хешування алгоритмом md5 рядка, що був зканкатенований на основі дати логіну, паролю та додаткових ключів.

Клас Email створений для обробки функціоналу пов’язаного з електронними листами. Коли надходить запит на відправку електронного листа, формується лист шляхом заповнення параметрів, таких як: хост, автентифікація, логін відправника, пароль відправника, шифрування тощо. Перед відправкою повідомлення, йде запит до функції getBody, яка в залежності від типу листа віддає потрібний шаблон повідомлення. Після

успішної відправки листа, додається запис до БД, з інформацію про відправлений лист і повертається на клієнтську частину програми відповідь «Message sent». У випадку, коли повідомлення не було успішно надіслано, до клієнта повертається помилка з повним описом проблеми. Для відправки електронних листів задіяно модуль PHPMailer, який попередньо завантажено, задля підвищення швидкодії, а також запобігання проблем, що можуть виникнути у разі оновлення модуля.

Для запуску серверної частини програми на сервері, треба перейменувати директорію з файлами у «mailer_api» та розмістити в корені директорії, що викликається при зверненні по доменному імені (на Linux хостингах – «public_html»). Наступним кроком буде створення пустої бази даних та користувача з усіма правами до неї на сервері, дані яких треба внести в змінні в файлі mailer_api/config/connection.php.

4.4 Очікувані техніко-економічні показники

Обране технічне рішення засноване на використанні React.js та PHP для створення веб-додатку з масовою розсилкою електронних листів. Це рішення вибране через свою ефективність, швидкість та гнучкість у розробці. React.js дозволяє створювати динамічні та зручні інтерфейси, забезпечуючи плавність роботи додатку. PHP, у свою чергу, забезпечує зручний доступ до бази даних та оптимальну обробку запитів.

Очікувані переваги вибору технічного рішення:

- Швидкість та Продуктивність: Використання React.js та PHP дозволяє створити додаток, що працює швидко та ефективно, забезпечуючи користувачам комфортне використання.
- Масштабованість: Обрані технології підтримують масштабованість додатку, що дозволяє йому рості з розвитком бізнесу або збільшенням обсягу користувачів.

- Легкість Розробки та Підтримки: React.js та PHP - це популярні технології з великою спільнотою розробників, що забезпечує доступ до багатьох ресурсів та готових рішень для розв'язання проблем.
- Економія Ресурсів: Обране технічне рішення відповідає мінімальним вимогам до обладнання, що дозволяє зберігати ресурси комп'ютерів та знижувати витрати на інфраструктуру.

5 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

5.1 Мета і завдання експерименту

Мета експерименту полягає в перевірці функціональності, ефективності та безпеки розробленого веб-додатку для масованої розсилки електронних листів. Завдання експерименту включає проведення ряду тестів для оцінки різних аспектів системи.

1. Функціональність та Інтерфейс:
   a. Перевірка інтуїтивно зрозумілого інтерфейсу для користувачів.
   b. Тестування налаштовування та відстеження розсилки через конструктор шаблонів.
   c. Визначення ефективності інструментів для імпорту та управління списками адресатів.
2. Персоналізація та Креативний Вміст:
   a. Використання тестових даних для персоналізації листів та вставки індивідуальних даних.
   b. Перевірка підтримки HTML-формату для створення креативного вмісту листів.
3. Безпека Даних:
   a. Аналіз використання шифрування для захисту інформації про адресатів та відправників.
   b. Тестування заходів безпеки для забезпечення конфіденційності особистих даних.
4. Оптимізація та Швидкодія:
   a. Проведення тестів на оптимізацію процесу розсилки через ефективні алгоритми та механізми кешування.
   b. Перевірка можливості налаштовування параметрів доставки, таких як інтервали розсилки.

Після проведення цих тестів, мета експерименту - забезпечити, що розроблений веб-додаток відповідає всім технічним вимогам, функціонує ефективно та забезпечує високий рівень безпеки для користувачів та їхніх даних.

5.2 Вимоги до експерименту

Вимоги до експерименту, що допоможуть визначити ефективність, стабільність та безпеку розробленого веб-додатку під час його реального функціонування та взаємодії з різними аспектами веб-середовища:

Тестування Навантаження:

Провести тестування системи на великому обсязі даних та визначити, як добре вона впорається з великою кількістю розсилок та адресатів.

Валідація Даних:

Перевірити валідацію введених даних користувачами для запобігання введенню некоректних чи шкідливих даних, які можуть впливати на безпеку системи.

Тестування Роботи з Різними Браузерами:

Провести тестування веб-додатку на різних веб-браузерах, включаючи Chrome, Firefox, Safari та Edge, для переконання в сумісності та однаковій ефективності.

Моніторинг Логів та Журналів:

Забезпечити належний моніторинг логів та журналів подій для вчасного виявлення та реагування на можливі проблеми, помилки чи загрози безпеки.

Тестування Стабільності:

Провести тести на стабільність системи, переконавшись, що вона не схильна до збоїв та має низький рівень відмов.

Тестування Масштабованості:

Оцінити можливість системи масштабуватися з ростом кількості користувачів та обсягу розсилок, забезпечуючи при цьому стабільну та продуктивну роботу.

5.3 Результати експерименту

Після проведення експерименту з масованою розсилкою електронних листів і виконання ряду тестів, отримано результати, що свідчать про високий рівень функціональності, ефективності та безпеки розробленого веб-додатку.

5.3.1 Функціональність та Інтерфейс

a. Інтуїтивний інтерфейс: Під час тестування підтверджено, що інтерфейс веб-додатку є інтуїтивно зрозумілим для користувачів. Всі основні функції легко доступні, що сприяє зручності користування.

b. Конструктор шаблонів: Тестування налаштовування та відстеження розсилки показало, що ці інструменти дозволяють користувачам ефективно створювати та відправляти персоналізовані листи без зайвих труднощів.

c. Імпорт та управління списками адресатів: Виявлено високу ефективність інструментів для імпорту та управління списками адресатів. Система легко справляється з великим обсягом даних.

5.3.2 Персоналізація

a. Персоналізація листів: Використання тестових даних підтвердило успішну персоналізацію листів та вставку індивідуальних даних.

b. HTML-формат: Тестування підтвердило підтримку HTML-формату для створення креативного та привабливого вмісту листів.

5.3.3 Безпека Даних

a. Шифрування: Аналіз підтвердив використання шифрування для захисту інформації про адресатів та відправників.

b. Заходи безпеки: Тестування підтвердило ефективність заходів безпеки для забезпечення конфіденційності особистих даних.

5.3.4 Оптимізація та Швидкодія

a. Оптимізація процесу розсилки: Тести підтвердили високу оптимізацію процесу розсилки за допомогою ефективних алгоритмів.

b. Параметри доставки: Вивчення параметрів доставки підтвердило можливість налаштовування інтервалів розсилки, що сприяє гнучкості використання системи.

5.3.5 Вимоги до експерименту

a. Тестування Навантаження: Система ефективно впоралася з великою кількістю розсилок та адресатів, демонструючи високий рівень масштабованості.

b. Валідація Даних: Валідація даних користувачами ефективно запобігала введенню некоректних чи шкідливих даних, забезпечуючи безпеку системи.

c. Тестування Роботи з Різними Браузерами: Веб-додаток показав сумісність та однакову ефективність на різних веб-браузерах, включаючи Chrome, Opera, Firefox та Edge.

d. Тестування Стабільності: Система виявилася стабільною, з низьким рівнем відмов та високою надійністю.

Ці результати підтверджують, що розроблений веб-додаток відповідає всім технічним вимогам, функціонує ефективно та забезпечує високий рівень безпеки для користувачів та їхніх даних.

ВИСНОВКИ

Кваліфікаційна робота є завершеною науковою роботою, в якій вирішена науково-практична задача розробки комп’ютерної системи контролю автоматизованої розсилки електронних повідомлень.

Під час виконання дипломної роботи було досліджено різні найпопулярніші середовища розробки веб додатків, фреймворки та бібліотеки для розробки клієнтської частини на мові JavaScript та мови для розробки серверної частини. На основі аналізу, було обрано середовище Visual Studio Code, через можливість підключення розширень, що збільшує можливість налаштування для зручнішого використання; для розробки клієнтської частини, було обрано React.js через «легкий вхід» для тих, хто володіє JavaScript; для розробки серверної частини, обрано мову PHP, через можливість легкого вбудування інформації в HTML розмітку.

В контексті розробки системи керування розсилкою електронних листів та визначення технічних вимог до неї, важливо враховувати широкий спектр аспектів, починаючи від зручного інтерфейсу користувача та закінчуючи ефективним функціоналом серверного обладнання.

Загальна концепція та технічні вимоги, визначені у тексті, вказують на необхідність комплексного підходу до розробки системи, який охоплює як користувацькі, так і технічні аспекти для забезпечення якості та ефективності функціонування системи керування розсилкою електронних листів.

В даній роботі було закріплено, поглиблено та узагальнено знання з розробки веб-додатків мовами програмування JavaScript та PHP, та їх фреймворками, а також розробки зручних інтерфейсів за допомогою Figma. Основні висновки і результати роботи полягають у наступному:

1. Досліджені існуючі системи контролю розсилки електронних листів та визначено їх переваги та недоліки.
2. Синтезовано структурну схему програми.

3. Розроблено сучасний користувацький інтерфейс за допомогою онлайн-сервісу Figma.

4. Розроблено та створено веб-додаток для керування автоматичною розсилкою електронних листів.

Результати роботи можуть бути корисними компаніям чи окремим користувачам в створенні автоматизованої розсилки електронних листів з відстеженням відкриття листа, відправкою в обраний час та конструктором шаблонів повідомлень.

ПЕРЕЛІК ПОСИЛАНЬ

1. https://mailchimp.com/;
2. https://www.zoho.com/campaigns/;
3. https://sendpulse.ua;
4. https://sendexpert.ua/emails;
5. https://nethunt.ua/blog/email-markietingh-v-2021-rotsi-iak-iefiektivno-robiti-email-rozsilki/;
6. https://www.resonatehq.com;
7. https://horoshop.ua/ua/blog/email-marketing/;
8. https://webprofit.com.ua/e-mail-marketing-dlya-biznesu/;
9. https://code.visualstudio.com/;
10. https://visualstudio.microsoft.com/;
11. https://marketplace.visualstudio.com/;
12. https://uk.wikipedia.org/wiki/PhpStorm;
13. https://uk.wikipedia.org/wiki/PhpStorm;
14. https://itpro.ua/product/jetbrains-phpstorm/?tab=description;
15. https://ua.vuejs.org/guide/introduction.html;
16. https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started;
17. https://react.dev/;
18. https://angular.io/;
19. https://dan-it.com.ua/blog/chto-jeto-takoe-node-js-prostymi-slovami/;
20. https://nodejs.org/uk;
21. https://itproger.com/ua/course/csharp;
22. https://www.php.net/.

ДОДАТОК А

Текст клієнтської частини програми

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

ПРОГРАМНЕ ЗАБЕЗБПЕЧЕННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ ІТ-КОМПАНІЇ З УРАХУВАННЯМ СЕРВІСУ КОНТРОЛЯ РОЗСИЛКИ ЕЛЕКТРОННИХ ЛИСТІВ

Текст клієнтської частини програми
804.02070743.23001-01 12 01
Листів 29

2023

src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.scss';
import App from './App';
import ToastContextProvider from './contexts/toastContext';
import AccountsContextProvider from './contexts/accountsContext';
import TokenContextProvider from './contexts/tokenContext';
import ModalContextProvider from './contexts/modalContext';
import RecipientsContextProvider from './contexts/recipientsContext';
import UserContextProvider from './contexts/userContext';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
 <UserContextProvider>
  <RecipientsContextProvider>
   <ModalContextProvider>
    <TokenContextProvider>
     <AccountsContextProvider>
      <ToastContextProvider>
       <App />
      </ToastContextProvider>
     </AccountsContextProvider>
    </TokenContextProvider>
   </ModalContextProvider>
  </RecipientsContextProvider>
 </UserContextProvider>
);
```

src/App.js

```
import { useContext, useEffect, useRef } from 'react';
import { BrowserRouter, Route, Routes } from 'react-router-dom';
import Nav from './components/controls/nav/Nav';
import Toast from './components/toast/Toast';
import { ToastContext } from './contexts/toastContext';
import { AccountsContext } from './contexts/accountsContext';
import ChangeData from './services/changeData';
import Enter from './components/enter/Enter';
import checkToken from './services/enter/checkToken';
import { TokenContext } from './contexts/tokenContext';
import Modal from './components/modal/Modal';
import { RecipientsContext } from './contexts/recipientsContext';
import PageTemplate from './components/pageTemplate/PageTemplate';
import { UserContext } from './contexts/userContext';

export default function App() {
  const checkInterval = 1500;
  const toLoginTimer = useRef(null);
  const { contextToast, setContextToast } = useContext(ToastContext);
  const { contextUser, setContextUser } = useContext(UserContext);
  const { setContextAccounts } = useContext(AccountsContext);
  const { contextToken, setContextToken } = useContext(TokenContext);
  const { setContextRecipients } = useContext(RecipientsContext);

  useEffect(() => {
    async function initialCheck() {
      if (localStorage.getItem('token') && localStorage.getItem('login')) {
```

```
            const data = await checkToken(setContextToast);

            setContextToken(data.status);
            setContextUser({
              id: data.id,
              login: data.login,
              messageTemplate: data.messageTemplate
            });
          }
        }
        initialCheck();

        toLoginTimer.current = setTimeout(() => {
          if (document.location.pathname !== '/') {
            document.location.pathname = '/';
          }
        }, checkInterval);
      }, []);

      useEffect(() => {
        if (contextToken === 'ok') {
          clearTimeout(toLoginTimer.current);
          ChangeData('receive', setContextAccounts, 'accounts', contextUser.id);
          ChangeData('receive', setContextRecipients, 'contacts', contextUser.id);
        }
      }, [contextToken, setContextToken]);

      return (
        <div className="App">
          <header>
            <h1>Mailer</h1>
          </header>

          <BrowserRouter>
            <Modal />
            <Nav />
            <Routes>
              <Route path="/" element={<Enter />} />

              <Route path="/:id" element={<PageTemplate />} />
            </Routes>
          </BrowserRouter>

          {contextToast.length ? (
            <div id="toasts">
              {contextToast.map((toast, i = 0) =>
                i < 5 ? <Toast info={toast} key={i} /> : ''
              )}
            </div>
          ) : (
            ''
          )}
        </div>
      );
    }
```

src/api/base-api.service.js

```
export default async function baseAPI(params, formData) {
  if (!formData) formData = new FormData();

  return fetch(`http://localhost/mailer_api/?${params}`, { method: 'POST', body: formData });
}

src/components/contactsList/ContactsList.jsx
import { useContext, useEffect, useMemo, useState } from 'react';
import { RecipientsContext } from '../../contexts/recipientsContext';
import ChangeData from '../../services/changeData';
import Search from '../search/Search';
import './ContactsList.scss';
import { UserContext } from '../../contexts/userContext';

export default function ContactsList({ type }) {

  const { contextUser, setContextUser } = useContext(UserContext);
  const { contextRecipients, setContextRecipients } = useContext(RecipientsContext);
  const [isCheckedAll, setIsCheckedAll] = useState(false);
  const [totalNumber, setTotalNumber] = useState(0);
  const [checkedNumber, setCheckedNumber] = useState(0);
  const [tasks, setTasks] = useState([]);
  const [messages, setMessages] = useState([]);
  const parent = type === 'sent' ? 'mailer' : type === 'inTask' ? 'mailer' : 'modal'

  const hiddenColumnsContacts = ['id', 'time', 'userId'];
  const hiddenColumnsTasks = ['id', 'userId'];
  const hiddenColumnsMessages = ['id', 'opened', 'userId'];

  const data = useMemo(
    () => (type === 'sent' ? messages : type === 'contacts' ? contextRecipients : type === 'inTask' ? tasks :
''),
    [messages, tasks, contextRecipients, type]
  );
  const hiddenColumns = useMemo(
    () => (type === 'sent' ? hiddenColumnsMessages : type === 'contacts' ? hiddenColumnsContacts : type
=== 'inTask' ? hiddenColumnsTasks : ''),
    [messages, tasks, contextRecipients, type]
  );


  function checkAll() {
    for (let input of document.querySelectorAll(`.${parent} .blackBox .row:not(.dNone)
input[type="checkbox"]`)) {
      input.checked = !isCheckedAll;
    }
    setIsCheckedAll(!isCheckedAll);
    handlerSetCheckedNumber();
  }

  function handlerSetCheckedNumber() {
    setCheckedNumber(document.querySelectorAll(`.${parent} .blackBox .row
input[type="checkbox"]:checked`).length);
  }

  function getIdS() {
    let tmp = [];
```

```
        document.querySelectorAll(`.${parent} .blackBox .row input[type="checkbox"]:checked`)
          .forEach(message => tmp.push(message.getAttribute('id')))
        return tmp;
      }

      function changeDateFormat(date) {
        return date.slice(3, 6) + date.slice(0, 3) + date.slice(6);
      }

      useEffect(() => {
        setTotalNumber(document.querySelectorAll(`.${parent} .blackBox .row:not(.dNone)
input[type="checkbox"]`).length);
        handlerSetCheckedNumber();
      }, [messages, tasks, parent])

      useEffect(() => {
        if (type === 'inTask' && contextUser.id) {
          ChangeData(
            'receive',
            setTasks,
            'tasks',
            contextUser.id
          );
        } else if (type === 'sent' && contextUser.id) {
          ChangeData(
            'receive',
            setMessages,
            'messages',
            contextUser.id
          );
        } else if (type === 'contacts' && contextUser.id) {
          ChangeData(
            'receive',
            setContextRecipients,
            'contacts',
            contextUser.id
          );
        }
      }, [type, contextUser.id])

      return (
        <form className='contactsList'>
          {
            data.length && type ?
              <>
                <div className="row contactsList_counter">
                  <p>Усього: {totalNumber}</p>
                  <p>Обрано: {checkedNumber}</p>
                </div>
                <div className="row contactsList_controls">
                  <p onClick={checkAll}>
                    {isCheckedAll ? 'Зняти виділення' : 'Обрати всі'}
                  </p>
                  {
                    checkedNumber > 0 && type === 'inTask' ?
                      <p onClick={() => ChangeData(
                        'delete',
```

```
                        setTasks,
                        'tasks',
                        contextUser.id,
                        getIdS()
                      )}>Видалити обрані</p>
                      : ''
                  }
                </div>
                <Search handlerChange={() => setTotalNumber(document.querySelectorAll(`.${parent}
.blackBox .row:not(.dNone) input[type="checkbox"]`).length)} />
                <div className="row blackBox_header">
                  {
                    Object.keys(data[0]).map((objectKey, n) =>
                      !hiddenColumns.includes(objectKey) ?
                        <p key={'header cell#' + n}>
                          {objectKey}
                        </p> :
                        ''
                    )
                  }

                </div>
                <div className='blackBox'>
                  {
                    data
                      .sort((a, b) => new Date(changeDateFormat(b.time)) - new
Date(changeDateFormat(a.time)))
                      .map(item =>
                        <div
                          key={'item#' + item.id}
                          className="row">
                          <input
                            type="checkbox"
                            id={`${item.id}`}
                            onChange={handlerSetCheckedNumber} />
                          <label htmlFor={`${item.id}`}></label>
                          {
                            item.opened ?
                              <svg
                                viewBox="0 0 512 512"
                                className='eye'>
                                <g>
                                  <path                                    d="M256,128c-81.9,0-145.7,48.8-
224,128c67.4,67.7,124,128,224,128c99.9,0,173.4-76.4,224-126.6   C428.2,198.6,354.8,128,256,128z M256,347.3c-
49.4,0-89.6-41-89.6-91.3c0-50.4,40.2-91.3,89.6-91.3s89.6,41,89.6,91.3      C345.6,306.4,305.4,347.3,256,347.3z"
/><g><path                          d="M256,224c0-7.9,2.9-15.1,7.6-20.7c-2.5-0.4-5-0.6-7.6-0.6c-28.8,0-52.3,23.9-
52.3,53.3c0,29.4,23.5,53.3,52.3,53.3                  s52.3-23.9,52.3-53.3c0-2.3-0.2-4.6-0.4-6.9c-5.5,4.3-12.3,6.9-
19.8,6.9C270.3,256,256,241.7,256,224z" /></g></g></svg>
                              : ''
                          }
                          {
                            Object.keys(item).map((objectKey, i) =>
                              !hiddenColumns.includes(objectKey) ?
                                <input
                                  key={'sent cell#' + item['id'] + i}
                                  type='text'
                                  name={objectKey}
```

```
                                    value={item[objectKey]}
                                    title={item[objectKey]}
                                    readOnly /> :
                                    ''
                                )
                            }
                        </div>
                    )
                }


                </div>
            </>
            : <p>Список пуст</p>
        }
    </form>
  )
}
```

src/components/controls/button/button.jsx

```
import './Button.scss';

export default function Button({ text, handlerClick }) {
  return <button onClick={handlerClick}>{text}</button>;
}
```

src/components/controls/nav/Nav.jsx

```
import { Link, useLocation } from 'react-router-dom';
import './Nav.scss';

export default function Nav() {
  const tabs = [
    {
      title: 'Пошта',
      link: '/mailer-->',
      subTabs: [
        ['write', 'Написати'],
        ['sent', 'Відправлені'],
        ['inTask', 'В черзі'],
      ],
    },
    {
      title: 'Налаштування',
      link: '/settings-->',
      subTabs: [
        ['profile', 'Профіль'],
        ['emails', 'Акаунти'],
        ['contacts', 'Контакти'],
      ],
    },
  ];

  let location = useLocation().pathname.split('/')[1];
  let chapter = location.split('--%3E')[0];
  let tab = location.split('--%3E')[1];

  return (
```

```
            <>
              {chapter !== '' ? (
                <nav>
                  <div className="top">
                    {tabs.map((chapter, i) => (
                      <div
                        className={`chapter ${chapter.subTabs.some((_subTabs) => _subTabs.includes(tab)) ?
'active' : ''}`}
                        key={i++}
                      >
                        {chapter.title}
                        <div className="subTabs">
                          {chapter.subTabs.map((subTab, s) => (
                            <Link
                              to={chapter.link + subTab[0]}
                              key={'subTab #' + s}
                              className={
                                subTab[0] === tab
                                  ? 'active'
                                  : ''
                              }
                            >
                              {subTab[1]}
                            </Link>
                          ))}
                        </div>
                      </div>
                    ))}
                  </div>

                  <div className="bottom">
                    <svg
                      viewBox="0 0 24 24"
                      className="logout"
                      onClick={() => {
                        localStorage.setItem('token', '');
                        window.location.pathname = '';
                      }}
                    >
                      <path d="M5 5v14a1 1 0 0 0 1 1h3v-2H7V6h2V4H6a1 1 0 0 0-1 1zm14.242-.97-8-2A1 1 0 0
0 10 3v18a.998.998 0 0 0 1.242.97l8-2A1 1 0 0 0 20 19V5a1 1 0 0 0-.758-.97zM15 12.188a1.001 1.001 0 0 1-2 0v-
.377a1 1 0 1 1 2 .001v.376z" />
                    </svg>
                  </div>
                </nav>
              ) : (
                ''
              )}
            </>
          );
        }
```

src/components/paginator/Paginator.jsx

```
import './Paginator.scss';

export default function Paginator({ pages, currentPage, setCurrentPage }) {
```

```
  return (
    <div className="row pagination">
      <div
        className={`leftArrow ${
          currentPage === 1 ? 'disabled' : ''
        }`}
        onClick={() => {
          if (currentPage > 1)
            setCurrentPage({
              currentPage: currentPage - 1,
              pagesNumber: pages
            });
        }}
      ></div>
      <p>{currentPage}</p>
      <div
        className={`rightArrow ${currentPage === Math.ceil(pages) ? 'disabled' : ''}`}
        onClick={() => {
          if (currentPage < Math.ceil(pages))
            setCurrentPage({
              currentPage: currentPage + 1,
              pagesNumber: pages
            });
        }}
      ></div>
    </div>
  )
}
```

src/components/enter/Enter.jsx

```
import { useContext, useState } from 'react';
import { ToastContext } from '../../contexts/toastContext';
import toLogIn from '../../services/enter/enter';
import Button from '../controls/button/Button';
import './Enter.scss';
import { UserContext } from '../../contexts/userContext';

export default function Enter() {
  const { contextToast, setContextToast } = useContext(ToastContext);
  const { contextUser, setContextUser } = useContext(UserContext);
  const [type, setType] = useState('login');

  function changeType(type) {
    setType();
    setTimeout(() => {
      setType(type);
    }, 200);
  }

  async function handlerSubmit(event) {
    const data = await toLogIn(event, contextToast, setContextToast, type);
    setContextUser({ id: data.id });
  }

  return (
    <div className={`enterForm ${type}`}>
      <form
```

```
  onSubmit={(event) => handlerSubmit(event)}
>
  {type === 'restore' || type === 'register' ? (
    <input
      type="email"
      name="email"
      placeholder="Email"
      required
    />
  ) : (
    ''
  )}

  {type === 'login' || type === 'register' ? (
    <>
      <input
        type="text"
        name="login"
        placeholder="Login"
        required
        onChange={({ target }) => target.value = target.value.replace(/[^A-z0-9]/g, '')}
      />
      <input
        type="password"
        name="password"
        placeholder="Password"
        required
      />
    </>
  ) : (
    ''
  )}

  {type === 'register' ? (
    <input
      type="password"
      name="confirmPassword"
      placeholder="Please confirm password"
      required
    />
  ) : (
    ''
  )}

  <Button
    text={
      type === 'login'
        ? 'Увійти'
        : type === 'register'
        ? 'Зареєструватися'
        : type === 'restore'
        ? 'Відновити пароль'
        : ''
    }
  />

  {
```

```
            <div className="row actions">
              {type === 'login' || type === 'register' ? (
                <p onClick={() => changeType('restore')}>
                  Відновити пароль
                </p>
              ) : (
                ''
              )}
              {type === 'restore' || type === 'register' ? (
                <p onClick={() => changeType('login')}>Увійти</p>
              ) : (
                ''
              )}
              {type === 'restore' || type === 'login' ? (
                <p onClick={() => changeType('register')}>
                  Зареєструватися
                </p>
              ) : (
                ''
              )}
            </div>
          }
        </form>
      </div>
    );
}
```

src/components/list/List.jsx

```
import { useEffect, useState } from 'react';
import './List.scss';

export default function List({ type, array = [] }) {

  const [currentElem, setCurrentElem] = useState({ name: 'Список пуст!' });

  useEffect(() => {
    if (array.length) {
      if (localStorage.getItem(type))
        setCurrentElem(array.find(elem => elem.id === localStorage.getItem(type)) || array[0])
      else
        setCurrentElem(array[0]);
    }
  }, [array])

  function handlerSelect(elem) {
    setCurrentElem(elem);
    localStorage.setItem(type, elem.id)
  }

  return (
    <div className='list' tabIndex='0'>
      {
        array && array.length && currentElem ?
          <>
            <div className="row list_currentElem">
              <span>{currentElem.email ? currentElem.email : ''}</span>
              <span>{currentElem.name}</span>
```

```
                        </div>
                        <div className="listBody">
                          <input type="hidden" name='selectedElem' value={currentElem.email ? currentElem.email
: currentElem.name} />
                          {
                            array.map(elem =>
                              <div className='row' key={'listRow#' + elem.id} onClick={() => handlerSelect(elem)}>
                                <span>{elem.email ? elem.email : ''}</span>
                                <span>{elem.name}</span>
                              </div>
                            )
                          }
                        </div>
                      </>
                      : 'Список пуст'
                  }
                </div>
              )
            }
```

src/components/modal/Modal.jsx

```
import { useContext, useEffect, useMemo, useState } from 'react';
import { ModalContext } from '../../contexts/modalContext';
import ChangeData from '../../services/changeData';
import ContactsList from '../contactsList/ContactsList';
import './Modal.scss';
import Button from '../controls/button/Button';
import { AccountsContext } from '../../contexts/accountsContext';
import { UserContext } from '../../contexts/userContext';
import { RecipientsContext } from '../../contexts/recipientsContext';

export default function Modal() {

  const [ title, setTitle ] = useState('');
  const [ inputs, setInputs ] = useState([]);
  const { contextUser, setContextUser } = useContext(UserContext);
  const { contextModal, setContextModal } = useContext(ModalContext);
  const { contextAccounts, setContextAccounts } = useContext(AccountsContext);
  const { contextRecipients, setContextRecipients } = useContext(RecipientsContext);
  const context = useMemo(
    () => (contextModal.tab === 'emails' ? contextAccounts : contextModal.tab === 'contacts' ?
contextRecipients : ''),
    [contextAccounts, contextRecipients, contextModal.tab]
  );
  const setContext = useMemo(
    () => (contextModal.tab === 'emails' ? setContextAccounts : contextModal.tab === 'contacts' ?
setContextRecipients : ''),
    [contextAccounts, contextRecipients, contextModal.tab]
  );

  useEffect(() => {
    switch (contextModal.tab) {
      case 'emails':
        setTitle('Новий акаунт');
        setInputs([
          { type: 'text', name: 'name', placeholder: "Ім'я", required: true },
          { type: 'text', name: 'smtp', placeholder: "smtp.gmail.com", required: true },
```

```
                    { type: 'email', name: 'email', placeholder: "Пошта", required: true },
                    { type: 'password', name: 'password', placeholder: "Пароль", required: true }
                ]);
                break;

            case 'contacts':
                setTitle('Новий контакт');
                setInputs([
                    { type: 'text', name: 'name', placeholder: "Ім'я", required: true },
                    { type: 'email', name: 'email', placeholder: "Пошта", required: true }
                ]);
                break;
        }
    }, [contextModal.tab, contextModal.show])

    function handlerSubmit(event) {
        event.preventDefault();
        ChangeData(
            'add',
            setContext,
            contextModal.tab === 'emails'
                ? 'accounts'
                : contextModal.tab,
            contextUser.id
        );
        setContextModal({show: false});
    }

    return (
        <div className={`modal ${contextModal.show ? 'show' : ''}`}>
            <svg
                viewBox="0 0 128 128"
                className="close"
                onClick={() => setContextModal({...contextModal, show: false })} >
                <path  d="M81.646,64l22.248-22.249c3.48-3.48,3.474-9.131-0.019-12.623l-5.006-5.005  c-3.489-3.49-9.142-3.499-12.622-0.019L64,46.354L41.753,24.106c-3.484-3.483-9.133-3.472-12.624,0.018l-5.005,5.005  c-3.491,3.492-3.501,9.14-0.018,12.623L46.354,64L24.108,86.246c-3.483,3.484-3.472,9.133,0.018,12.623l5.005,5.006
c3.492,3.492,9.14,3.502,12.623,0.018L64,81.647l22.247,22.246c3.48,3.481,9.131,3.475,12.622-0.019l5.006-5.006
c3.489-3.489,3.498-9.142,0.019-12.622L81.646,64z" />
            </svg>
            {contextModal.type === 'contacts' ?
                <div className="container">
                    <ContactsList type='contacts' />
                </div>
                : (
                <div className={`container ${contextModal.tab}`}>
                    <form onSubmit={(event) => handlerSubmit(event)} >
                        <h3>{title}</h3>
                        {
                            inputs.length ? inputs.map((input, i) =>
                                <input
                                    key={contextModal.tab + ' input#' + i}
                                    type={input.type}
                                    name={input.name}
                                    placeholder={input.placeholder}
                                    required={input.required} />
```

```
                ) : ''
              }
              <Button
                className="add"
                text="Додати" />
            </form>
          </div>
        )}
      </div>
    );
}
```

src/components/pageTemplate/mailer/Mailer.jsx

```
import { useContext, useEffect, useState } from 'react';
import { AccountsContext } from '../../../contexts/accountsContext';
import { RecipientsContext } from '../../../contexts/recipientsContext';
import { ModalContext } from '../../../contexts/modalContext';
import { ToastContext } from '../../../contexts/toastContext';
import PreparationForSend from '../../../services/emailSending/PreparationForSend';
import ReadFile from '../../../services/emailSending/ReadFile';
import Button from '../../controls/button/Button';
import List from '../../list/List';
import './Mailer.scss';
import ContactsList from '../../contactsList/ContactsList';
import { UserContext } from '../../../contexts/userContext';

export default function Mailer({ tab }) {
  const { contextRecipients, setContextRecipients } = useContext(RecipientsContext);
  const { contextToast, setContextToast } = useContext(ToastContext);
  const { contextModal, setContextModal } = useContext(ModalContext);
  const { contextUser, setContextUser } = useContext(UserContext);
  const { contextAccounts } = useContext(AccountsContext);
  const [errors, setErrors] = useState([]);

  return (
    <div className="mailer">
      {tab === 'write' ? (
        <form
          onSubmit={(event) =>
            PreparationForSend(
              event,
              contextUser.id,
              contextToast,
              setContextToast,
              contextAccounts,
              errors,
              setErrors
            )
          }
        >
          {contextAccounts.length && contextUser.messageTemplate ? (
            <>
              <List
                type="accounts"
                array={contextAccounts} />
              <div className="row">
                <input
```

```
                        type="file"
                        name="file"
                        accept=".csv"
                        hidden
                        onChange={({ target }) =>
                          ReadFile(
                            contextUser.id,
                            target.files[0],
                            setContextRecipients,
                            contextToast,
                            setContextToast
                          )
                        } />
                      <Button
                        text="Обрати файл з контактами"
                        handlerClick={(event) => {
                          event.preventDefault();
                          event.target.previousElementSibling.value = '';
                          event.target.previousElementSibling.click();
                        }} />
                      <Button
                        text="Обрати контакти з списку"
                        handlerClick={(event) => {
                          event.preventDefault();
                          setContextModal({
                            ...contextModal,
                            type: 'contacts',
                            show: true
                          });
                        }} />
                    </div>
                    <div className="row">
                      <input type="date" name="date" />
                      <input
                        type="time"
                        name="time"
                        className="time"
                      />
                    </div>
                    <Button text="Надіслати" />

                    <div className="errorsWindow">
                      {
                      errors.length
                        ? errors.map((error, k) => (
                            <p key={k++}>{error}</p>
                          ))
                        : ''
                      }
                    </div>
                  </>
                ) : <p>Для початку роботи треба додати пошту та створити шаблон повідомлення. Це можна зробити в налаштуваннях.</p>
              }
            </form>
          ) : tab === 'sent' || tab === 'inTask' ?
            <ContactsList type={tab} />
```

```
        : ''}
      </div>
    );
}
```

src/components/pageTemplate/settings/Settings.jsx

```
import { useContext, useEffect, useState } from 'react';
import { AccountsContext } from '../../../contexts/accountsContext';
import { RecipientsContext } from '../../../contexts/recipientsContext';
import { ModalContext } from '../../../contexts/modalContext';
import { ToastContext } from '../../../contexts/toastContext';
import PreparationForSend from '../../../services/emailSending/PreparationForSend';
import ReadFile from '../../../services/emailSending/ReadFile';
import Button from '../../controls/button/Button';
import List from '../../list/List';
import './Mailer.scss';
import ContactsList from '../../contactsList/ContactsList';
import { UserContext } from '../../../contexts/userContext';

export default function Mailer({ tab }) {
  const { contextRecipients, setContextRecipients } = useContext(RecipientsContext);
  const { contextToast, setContextToast } = useContext(ToastContext);
  const { contextModal, setContextModal } = useContext(ModalContext);
  const { contextUser, setContextUser } = useContext(UserContext);
  const { contextAccounts } = useContext(AccountsContext);
  const [errors, setErrors] = useState([]);

  return (
    <div className="mailer">
      {tab === 'write' ? (
        <form
          onSubmit={(event) =>
            PreparationForSend(
              event,
              contextUser.id,
              contextToast,
              setContextToast,
              contextAccounts,
              errors,
              setErrors
            )
          }
        >
          {contextAccounts.length && contextUser.messageTemplate ? (
            <>
              <List
                type="accounts"
                array={contextAccounts} />
              <div className="row">
                <input
                  type="file"
                  name="file"
                  accept=".csv"
                  hidden
                  onChange={({ target }) =>
                    ReadFile(
                      contextUser.id,
```

```
                                target.files[0],
                                setContextRecipients,
                                contextToast,
                                setContextToast
                              )
                            } />
                          <Button
                            text="Обрати файл з контактами"
                            handlerClick={(event) => {
                              event.preventDefault();
                              event.target.previousElementSibling.value = '';
                              event.target.previousElementSibling.click();
                            }} />
                          <Button
                            text="Обрати контакти з списку"
                            handlerClick={(event) => {
                              event.preventDefault();
                              setContextModal({
                                ...contextModal,
                                type: 'contacts',
                                show: true
                              });
                            }} />
                        </div>
                        <div className="row">
                          <input type="date" name="date" />
                          <input
                            type="time"
                            name="time"
                            className="time"
                          />
                        </div>
                        <Button text="Надіслати" />

                        <div className="errorsWindow">
                          {
                          errors.length
                            ? errors.map((error, k) => (
                                <p key={k++}>{error}</p>
                              ))
                            : ''
                          }
                        </div>
                      </>
                    ) : <p>Для початку роботи треба додати пошту та створити шаблон повідомлення. Це можна зробити в налаштуваннях.</p>
                    }
                  </form>
                ) : tab === 'sent' || tab === 'inTask' ?
                  <ContactsList type={tab} />
                : ''}
              </div>
            );
          }

          src/components/pageTemplate/PaheTemplate.jsx
          import { useParams } from 'react-router-dom';
```

```
import Mailer from './mailer/Mailer';
import './PageTemplate.scss';
import Settings from './settings/Settings';

export default function PageTemplate() {
  const { id } = useParams();

  let chapter = id.split('-->')[0];
  let tab = id.split('-->')[1];

  return (
    <main className="container pageTemplate">
      {chapter === 'mailer' ? (
        <Mailer tab={tab} />
      ) : chapter === 'settings' ? (
        <Settings tab={tab} />
      ) : (
        ''
      )}
    </main>
  );
}
```

src/components/search/Search.js

```
import './Search.scss';

export default function Search({ handlerChange }) {

  function Search(input) {
    let rows = input.nextElementSibling.nextElementSibling.children;

    if (input.value.length > 2)
      for (let row of rows) {
        for (let cell of row.children) {
          if (cell.tagName === 'INPUT' && cell.value.toLowerCase().includes(input.value.toLowerCase()))
{
            row.classList.remove('dNone');
            break
          }
          else
            row.classList.add('dNone');
        }
      }
    else
      for (let row of rows)
        row.classList.remove('dNone');

    handlerChange();
  }

  return (
    <input type="text" className='search' placeholder='Пошук, від 3х символів' onChange={({ target })
=> Search(target)} />
  )
}
```

src/components/table/Table.jsx

```
import { UserContext } from '../../contexts/userContext';
import ChangeData from '../../services/changeData';
import Paginator from '../controls/paginator/Paginator';
import './Table.scss';
import React, { useContext, useEffect, useState } from 'react';

export default function Table({ tab, data, setData }) {

  const { contextUser, setContextUser } = useContext(UserContext);
  const [paginator, setPaginator] = useState({
    currentPage: 1,
    pagesNumber: 1,
  });

  let table = '';

  switch (tab) {
    case 'emails':
      table = 'акаунтів';
      break;

    case 'contacts':
      table = 'контактів';
      break;
  }

  useEffect(() => {
    setPaginator({ ...paginator, currentPage: 1 });

    if (data && data.length > 1) {
      setPaginator({ ...paginator, pagesNumber: data.length / 10 });
    } else {
      setPaginator({ ...paginator, pagesNumber: 1 });
    }
  }, [data, tab]);

  return (
    <>
      <Paginator pages={paginator.pagesNumber} currentPage={paginator.currentPage}
setCurrentPage={setPaginator} />

      <table className='customTable'>
        <tbody>
          {
            data.length ? data.map((elem, i) =>
              <React.Fragment key={'row#' + i}>
                {
                  i === 0 ?
                    <tr>
                      {
                        Object.keys(elem).map((objectKey, n) =>
                          objectKey !== 'id' && objectKey !== 'userId' && objectKey !== 'opened' &&
objectKey !== 'time' ?

                            <td key={'column#' + n}>
                              {objectKey}
                            </td>
                          : ''
```

```
                                    )}
                                </tr>
                            : ''
                        }
                        {
                            i >= (paginator.currentPage - 1) * 10 && i < (paginator.currentPage - 1) * 10 + 10 ?
                                <tr>
                                    {Object.keys(elem).map(
                                        (objectKey, k) =>
                                            objectKey !== 'id' && objectKey !== 'userId' && objectKey !== 'opened' &&
objectKey !== 'time' ?
                                                <td
                                                    key={`row#${i}cell#${k}`}
                                                    className={objectKey === 'password' ? 'password' : ''} >
                                                    {
                                                        objectKey === 'image' ?
                                                            <img
                                                                src={elem[objectKey]}
                                                                alt="" />
                                                            : elem[objectKey]
                                                    }
                                                </td> : ''
                                    )}
                                    <td
                                        className="controls"
                                        key={'row#' + i + 'controls'}>
                                        <button
                                            className="delete"
                                            onClick={() => ChangeData(
                                                'delete',
                                                setData,
                                                tab === 'emails' ? 'accounts' : tab,
                                                contextUser.id,
                                                [elem['id']]
                                            )
                                            }>
                                            <svg viewBox="0 0 32 32">
                                                <path
                                                    d="M27,6h-6V5c0-1.654-1.346-3-3-3h-4c-1.654,0-3,1.346-
3,3v1H5C3.897,6,3,6.897,3,8v1c0,0.552,0.448,1,1,1h24  c0.552,0,1-0.448,1-1V8C29,6.897,28.103,6,27,6z M13,5c0-
0.551,0.449-1,1-1h4c0.551,0,1,0.449,1,1v1h-6V5z"
                                                    id="XMLID_246_"/>
                                                <path      d="M6,12v15c0,1.654,1.346,3,3,3h14c1.654,0,3-1.346,3-3V12H6z
M19.707,22.293        c0.391,0.391,0.391,1.023,0,1.414s-1.023,0.391-1.414,0L16,21.414l-2.293,2.293c-0.391,0.391-
1.023,0.391-1.414,0          s-0.391-1.023,0-1.414L14.586,20l-2.293-2.293c-0.391-0.391-0.391-1.023,0-1.414s1.023-
0.391,1.414,0L16,18.586l2.293-2.293                                                    c0.391-0.391,1.023-
0.391,1.414,0s0.391,1.023,0,1.414L17.414,20L19.707,22.293z" />
                                            </svg>
                                        </button>
                                    </td>
                                </tr>
                            : ''
                        }
                    </React.Fragment>
                ) : <tr>
                    <td colSpan="5">
                        Таблиця {table} пуста!
```

```
                    </td>
                  </tr>
                }
              </tbody>
            </table>
          </>
        )
      }
```

src/components/toast/Toast.jsx

```
import './Toast.scss';
import { useContext} from 'react';
import { ToastContext } from '../../contexts/toastContext';

export default function Toast({ info }) {

  const { contextToast, setContextToast } = useContext(ToastContext);

  const handlerClose = () => {
    let tmp = contextToast.filter(toast => toast !== info);
    setContextToast(tmp);
  }

  return (
    <div className={`toast ${info.type}`}>
      <svg viewBox="0 0 512 512" className='close' onClick={handlerClose}><path
d="M443.6,387.1L312.4,255.4l131.5-130c5.4-5.4,5.4-14.2,0-19.6l-37.4-37.6c-2.6-2.6-6.1-4-9.8-4c-3.7,0-7.2,1.5-
9.8,4 L256,197.8L124.9,68.3c-2.6-2.6-6.1-4-9.8-4c-3.7,0-7.2,1.5-9.8,4L68,105.9c-5.4,5.4-
5.4,14.2,0,19.6l131.5,130L68.4,387.1 c-2.6,2.6-4.1,6.1-
4.1,9.8c0,3.7,1.4,7.2,4.1,9.8l37.4,37.6c2.7,2.7,6.2,4.1,9.8,4.1c3.5,0,7.1-1.3,9.8-4.1L256,313.1l130.7,131.1
c2.7,2.7,6.2,4.1,9.8,4.1c3.5,0,7.1-1.3,9.8-4.1l37.4-37.6c2.6-2.6,4.1-6.1,4.1-
9.8C447.7,393.2,446.2,389.7,443.6,387.1z" /></svg>
      {info.txt}
      <span className='time'>{info.time}</span>
    </div>
  )
}
```

src/contexts/accountsContext.jsx

```
import React, { useState } from "react";

export const AccountsContext = React.createContext(null);

export default function AccountsContextProvider({ children }) {

  const [contextAccounts, setContextAccounts] = useState([]);

  return (
    <AccountsContext.Provider value={{
      contextAccounts, setContextAccounts
    }} >
      {children}
    </AccountsContext.Provider>
  );
}
```

src/contexts/modalContext.jsx

```
import React, { useState } from "react";

export const ModalContext = React.createContext(null);

export default function ModalContextProvider({ children }) {

   const [contextModal, setContextModal] = useState({show: false, type: ''});

   return (
      <ModalContext.Provider value={{
         contextModal, setContextModal
      }} >
         {children}
      </ModalContext.Provider>
   );
}
```

src/contexts/recipientsContext.jsx

```
import React, { useState } from "react";

export const RecipientsContext = React.createContext(null);

export default function RecipientsContextProvider({ children }) {

   const [contextRecipients, setContextRecipients] = useState([]);

   return (
      <RecipientsContext.Provider value={{
         contextRecipients, setContextRecipients
      }} >
         {children}
      </RecipientsContext.Provider>
   );
}
```

src/contexts/toastContext.jsx

```
import React, { useState } from "react";

export const ToastContext = React.createContext(null);
export default function ToastContextProvider({ children }) {

   const [contextToast, setContextToast] = useState([]);

   return (
      <ToastContext.Provider value={{
         contextToast, setContextToast
      }} >
         {children}
      </ToastContext.Provider>
   );
}
```

src/contexts/tokenContext.jsx

```
import React, { useState } from "react";

export const TokenContext = React.createContext(null);
```

```
export default function TokenContextProvider({ children }) {

  const [contextToken, setContextToken] = useState();

  return (
    <TokenContext.Provider value={{
      contextToken, setContextToken
    }} >
      {children}
    </TokenContext.Provider>
  );
}
```

src/contexts/userContext.jsx

```
import React, { useState } from "react";

export const UserContext = React.createContext(null);

export default function UserContextProvider({ children }) {

  const [contextUser, setContextUser] = useState({});

  return (
    <UserContext.Provider value={{ contextUser, setContextUser }} >
      {children}
    </UserContext.Provider>
  );
}
```

src/services/emailSending/afterSend.js

```
import GetTime from "../GetTime";

export default function afterSend(contextToast, setContextToast, type) {
  if (type === 'error')
    setContextToast([...contextToast, { type: 'error', txt: "Помилка! Спробуйте будьласка пізніше", time: GetTime('+3') }]);
  else {
    setContextToast([...contextToast, { type: 'success', txt: "Дані оброблені", time: GetTime('+3') }]);
  }
}
```

src/services/emailSending/DefineSeparator.js

```
export default function DefineSeparator(string) {
  if (string.indexOf(',') > 0)
    return ',';
  else if (string.indexOf(';') > 0)
    return ';';
}
```

src/services/emailSending/PreparationForSend.js

```
import GetTime from "../GetTime";
import afterSend from "./afterSend";
import baseAPI from "../../api/base-api.service";

export default async function PreparationForSend(event, userId, contextToast, setContextToast, contextAccounts, errors, setErrors) {

  event.preventDefault();
```

```
            let form = event.target,
                tmp_errors = [],
                recipients = [],
                request = 'send=true&userId=' + userId;

            function checkLength(date){
                return date.toString().length === 1 ? '0' + date : date;
            }

            function checkFinall(){
                if (counter === recipients.length) {
                    setErrors([...errors, ...tmp_errors]);
                    afterSend(contextToast, setContextToast);
                }
            }

            document.querySelectorAll('.modal input[type="checkbox"]:checked').forEach(input => {
                let tmp = [];
                let inputs = input.parentElement.children;
                for (let i = 3; i < inputs.length; i++)
                    if (inputs[i].name === 'id' || inputs[i].name === 'email' || inputs[i].name === 'name')
                        tmp.push(inputs[i].value);
                recipients.push(tmp);
            })

            if(!recipients.length){
                setContextToast([...contextToast, { type: 'inProgress', txt: "Оберіть отримувачів", time: GetTime('+3')
}]);
                return;
            }

            let date = document.querySelector('input[name="date"]').value;
            let time = document.querySelector('.time').value;
            if(!date && time) {
                let newDate = new Date();
                date = newDate.getFullYear() + '-' + parseInt(newDate.getMonth() + 1) + '-' + newDate.getDate();
            }

            let currentTime = new Date(date + ',' + time);
            if (currentTime > new Date()) {
                request = 'add=tasks&time=' + checkLength(currentTime.getDate()) + '.' +
checkLength(currentTime.getMonth() + 1) + '.' + currentTime.getFullYear() + ' ' +
checkLength(currentTime.getHours()) + ':' + checkLength(currentTime.getMinutes()) + '&userId=' + userId;
            }

            setContextToast([...contextToast, { type: 'inProgress', txt: "Дані обробляються", time: GetTime('+3') }]);

            let formData = new FormData(event.target);
            const account = contextAccounts.find(account => account.email === form.selectedElem.value);

            let counter = 0;
            for (let i = 0; i < recipients.length; i++) {
                formData.append('options', JSON.stringify({ senderId: userId, senderEmail: account.email,
senderPassword: account.password, smtp: account.smtp, recipientEmail: recipients[i][0] }));

                baseAPI(request, formData, 'mail')
                    .then((data) => {
```

```
                counter++;
                if (data.ok)
                    return data.json();
            })
            .then(data => {
                checkFinall();

                if (data !== 'Message sent' && data !== 'added')
                    tmp_errors.push(GetTime('+3') + ' ' + data);
            })
            .catch((data) => {
                checkFinall();
                return formData.get('options').split('recipientEmail":"')[1].split('","recipientName"')[0];
            });
        }
    }

    src/services/emailSending/ReadFile.js
    import ChangeData from "../changeData";
    import ShowToast from "../showToast";
    import DefineSeparator from "./DefineSeparator";

    export default function ReadFile(id, file, setContextRecipients, contextToast, setContextToast) {
        const reader = new FileReader();
        reader.readAsText(file);

        reader.onload = function () {
            let contacts = reader.result.split('\r\n'),
                separator = DefineSeparator(contacts[0]),
                values,
                emailColumnIndex,
                nameColumnIndex,
                data = [],
                columns = [
                    ['email', emailColumnIndex],
                    ['name', nameColumnIndex]
                ];

            for (let i = 0; i < contacts.length; i++) {
                values = contacts[i].split(separator);

                if (i === 0) {
                    for (let n = 0; n < values.length; n++)
                        for (let c = 0; c < columns.length; c++)
                            if (values[n].toLowerCase() === columns[c][0])
                                columns[c][1] = n;
                    for (let c = 0; c < columns.length; c++)
                        if (columns[c][1] === undefined) {
                            return ShowToast(contextToast, setContextToast, 'error', `Произошла ошибка, нет колонки
'${columns[c][0]}'`);
                        }
                }
                else {
                    if (!values.length || !values[0]) {
                        break;
                    }
```

```
            if (values[columns[1][1]].includes("'"))
              values[columns[1][1]] = values[columns[1][1]].replaceAll("'", "’")

            data.push([values[columns[1][1]], values[columns[0][1]]]);
          }
        }

        ShowToast(contextToast, setContextToast, 'success', 'Файл загружен');
        ChangeData('add', setContextRecipients, 'contacts', id, data);
      };
    }

    src/services/enter/checkToken.js
    import baseAPI from '../../api/base-api.service';

    export default async function checkToken(setContextToast) {
      try {
        const response = await baseAPI(
          'checkToken=' +
            localStorage.getItem('token') +
            '&login=' +
            localStorage.getItem('login')
        );

        const data = await response.json();

        if (data.status === 'ok') {
          if (document.location.pathname === '/') {
            document.location.pathname = '/mailer-->write';
          }
        } else {
          if (document.location.pathname !== '/')
            document.location.pathname = '/';
          else if (data.status === 'overdue payment') {
            setContextToast([
              {
                type: 'error',
                txt: 'Строк вашого сервісу сплинув. Будьласка поновіть оплату',
              },
            ]);
          }
        }

        return data;
      } catch (error) {
        console.error('Error in checkToken:', error);
      }
    }

    src/services/enter/enter.js
    import GetTime from '../GetTime';
    import baseAPI from '../../api/base-api.service';

    export default async function toLogIn(
      event,
      contextToast,
      setContextToast,
```

```
  type
) {
  event.preventDefault();

  let form = event.target;
  const formData = new FormData(form);

  if (type === 'login' || type === 'register') {
    if (form.login.value.length > 30 || form.login.value.length < 5) {
      setContextToast([
        { type: 'error', txt: 'Некоректна довжина поля' },
      ]);
      form.login.classList.add('error');
      return;
    } else form.login.classList.remove('error');

    if (form.password.value.length > 30 || form.password.value.length < 6) {
      setContextToast([
        { type: 'error', txt: 'Некоректна довжина поля' },
      ]);
      form.password.classList.add('error');
      return;
    } else form.password.classList.remove('error');
  }

  if (type === 'register') {
    if (form.confirmPassword.value !== form.password.value) {
      setContextToast([
        { type: 'error', txt: 'Некоректне підтвердження пароля' },
      ]);
      form.confirmPassword.classList.add('error');
      return;
    } else form.confirmPassword.classList.remove('error');
  }

  formData.append(
    'options',
    JSON.stringify([
      form.email?.value.toLowerCase() || '',
      form.login?.value || '',
      form.password?.value || ''
    ])
  );

  try {
    const response = await baseAPI(
      type + `=${type === 'login' ? form.login.value : 'true'}`,
      formData
    );

    if (response.ok) {
      const data = await response.json();

      if (data === 'no such user') {
        setContextToast([
          ...contextToast,
          {
```

```
                        type: 'error',
                        txt: 'Невірний логін або пароль',
                        time: GetTime('+3'),
                    },
                ]);
            } else if (data === 'no user with such email') {
                setContextToast([
                    ...contextToast,
                    {
                        type: 'error',
                        txt: 'Немає користувача з такою поштою',
                        time: GetTime('+3'),
                    },
                ]);
            } else if (data === 'restored') {
                setContextToast([
                    ...contextToast,
                    {
                        type: 'success',
                        txt: 'На пошту відправлено листа для відновлення пароля',
                        time: GetTime('+3'),
                    },
                ]);
            } else if (data === 'try later') {
                setContextToast([
                    ...contextToast,
                    {
                        type: 'error',
                        txt: 'Спробуйте будьласка пізніше',
                        time: GetTime('+3'),
                    },
                ]);
            } else if (data === 'added') {
                setContextToast([
                    ...contextToast,
                    {
                        type: 'success',
                        txt: 'Акаунт успішно створено. На пошту відправлено листа для підтвердження пошти.',
                        time: GetTime('+3'),
                    },
                ]);
            } else if (data === 'need verification') {
                setContextToast([
                    ...contextToast,
                    {
                        type: 'error',
                        txt: 'Пошту не підтверджено. Будьласка підтвердіть пошту у листі, що було надіслано на
вашу пошту.',
                        time: GetTime('+3'),
                    },
                ]);
            } else if (data === 'double') {
                setContextToast([
                    ...contextToast,
                    {
                        type: 'error',
                        txt: 'Такий логін чи пошта вже використовуються. Спробуйте інші дані.',
```

```
                    time: GetTime('+3'),
                  },
                ]);
              } else  {
                localStorage.setItem('login', data.login);
                localStorage.setItem('token', data.token);

                if (document.location.pathname === '/')
                  document.location.pathname = '/mailer-->write';

                return data;
              }
            } else {
              console.error('Non-OK response:', response);
            }
          } catch (error) {
            console.error('Error in performRequest:', error);
          }
        }

        src/services/changeData.js
        import baseAPI from "../api/base-api.service";

        export default async function ChangeData(type, setContext, table, userId, data = [''], column, value) {

          let operation = '', formData;

          data.forEach(item => {
            switch (type) {
              case 'receive':
              case 'add':
                operation = type + '=' + table;
                break;

              case 'edit':
                operation = 'table=' + table + '&edit=' + item + '&column=' + column + '&value=' + value;
                break;

              case 'delete':
                operation = 'delete=' + item + '&table=' + table;
                break;
            }

            if (userId) {
              operation += '&userId=' + userId;
            }

            if (type === 'add') {
              formData = new FormData(document.querySelector(`form`));

              if (!data[0]) {
                item = [];

                for (let input of document.querySelectorAll(`.${table === 'accounts' ? 'emails' : table} form
input`)) {
                  if (input.value.trim().length < 2 && input.getAttribute('name') !== 'selectedElem') {
                    input.classList.add('error');
```

```
                return;
              } else {
                input.classList.remove('error');

                if (input.name === 'file') {
                  formData.append('file', JSON.stringify(input.value));
                } else {
                  item.push(input.value);
                }
              }
            }
          }

          formData.append('options', JSON.stringify(item));
        }

        return baseAPI(operation, formData ? formData : '')
          .then((data) => {
            if (data.ok) return data.json()
          })
          .then((data) => {
            if (type === 'receive'){
              if (data === 'empty') {
                data = [];
              }

              setContext(data);
            } else if (type === 'delete' || type === 'edit' || type === 'add') {
              ChangeData('receive', setContext, table, userId);
            }
          });
    })
}

src/services/GetTime.js
export default function GetTime(utc) {
    const date = new Date();
    let tmp = date.getUTCHours() + +utc;
    const minutes = date.getMinutes().toString().length === 1 ? '0' + date.getMinutes() : date.getMinutes();
    const hours = tmp.toString().length === 1 ? '0' + tmp : tmp;

    return hours + ':' + minutes;
}

src/services/showToast.js
import GetTime from "./GetTime";

export default function ShowToast(context, setContext, type, txt){
    setContext([...context, { type: type, txt: txt, time: GetTime('+3') }]);
}
```

ДОДАТОК Б

Текст серверної частини програми

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

ПРОГРАМНЕ ЗАБЕЗБПЕЧЕННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ ІТ-КОМПАНІЇ З УРАХУВАННЯМ СЕРВІСУ КОНТРОЛЯ РОЗСИЛКИ ЕЛЕКТРОННИХ ЛИСТІВ

Текст серверної частини програми
804.02070743.23001-01 12 01
Листів 10

2023

index.php

```
<?php

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");

include_once './config/connection.php';
include_once './services/PHPMailer/PHPMailer.php';
include_once './services/PHPMailer/SMTP.php';
include_once './services/PHPMailer/Exception.php';
include_once './services/loadDataFromDB.php';
include_once './services/toProcessTasks.php';
include_once './services/hashPassword.php';
include_once './objects/data.php';
include_once './objects/email.php';
include_once './objects/login.php';

$database = new Database();
$db = $database->getConnection();
$data = new Data($db);
$login = new Login($db);
$email = new Email();

$result;
$queryType = 'other';

if (!empty($_GET['register'])) {
  $queryType = 'register';
} else if (!empty($_GET['login']) || !empty($_GET['checkToken'])) {
  $queryType = 'login';
} else if (!empty($_GET['restore'])) {
  $queryType = 'restore';
}  else  if  (!empty($_GET['verifyEmail'])  ||  !empty($_GET['mailOpened'])  ||
!empty($_GET['toProcessTasks'])) {
  $queryType = 'outside';
}

switch ($queryType) {
  case 'register':
    $result = $data->addData('users', '', json_decode($_POST['options']));
    break;

  case 'login':
    $userLogin = $_GET['login'];
    $user = $db->query("SELECT * FROM `users` WHERE `login` = '" . $userLogin . "'")-
>fetchAll(PDO::FETCH_ASSOC)[0] ?? [];

    if (!empty($user)) {
      if ($user['emailVerification'] !== '1') {
        $result = 'need verification';
      } else if (date('Y-m-d') > $user['endOfPayment']) {
        $result = 'overdue payment';
      } else if (!empty($_GET['login']) && !empty($_POST["options"])) {
        $result = $login->checkUser(json_decode($_POST["options"]));
      } else if (!empty($_GET['checkToken']) && !empty($_GET['login'])) {
        $result = $login->checkToken($_GET["checkToken"], $_GET["login"]);
      }
```

```
            } else $result = 'no such user';
            break;

        case 'restore':
            $_POST['options'] = '["kbylbas.itbest@gmail.com","",""]';
            $result = $data->restorePassword(json_decode($_POST['options']));
            break;

        case 'outside':
            if (!empty($_GET['verifyEmail'])) {
                $data->editData('users', $_GET['verifyEmail'], 'emailVerification', 1);
            } else if (!empty($_GET['mailOpened'])) {
                $data->editData('messages', $_GET['mailOpened'], 'opened', '1');
            } else if (!empty($_GET['toProcessTasks'])) {
                toProcessTasks();
            }
            break;

        default:
            if (!empty($_GET["send"])) {
                $emailData = json_decode($_POST["options"]);
                $result = $email->sendMail($_GET['userId'], $emailData, 'userTemplate');
            } else if (!empty($_GET['receive'])) {
                $result = $data->getData($_GET['receive'], $_GET['userId'] ?? '', $_GET['receive'] === 'tasks' ?
'future' : '');
            } else if (!empty($_GET['delete'])) {
                $result = $data->deleteData($_GET['table'], $_GET['delete']);
            } else if (!empty($_GET['add'])) {
                $result = $data->addData($_GET['add'], $_GET['userId'] ?? '', json_decode($_POST['options']),
$_GET['time'] ?? '');
            } else if (!empty($_GET['edit'])) {
                $result = $data->editData($_GET['table'], $_GET['edit'], $_GET['column'], $_GET['value']);
            } else if (!empty($_GET['changeProfile'])) {
                $result = $data->editData('users', $_GET['changeProfile'], $_GET['column'], $_GET['value']);
            }
            break;
    }

    if (!empty($result)) {
        echo json_encode($result);
    }

    config/connection.php
    <?php

    class Database
    {
        private $host = "localhost";
        private $db_name = "diploma";
        private $username = "root";
        private $password = "";

        public function getConnection()
        {
            return new PDO("mysql:host=" . $this->host . ";dbname=" . $this->db_name, $this->username, $this-
>password);
        }
```

```
}

objects/data.php
<?php

date_default_timezone_set('Europe/Kyiv');

class Data
{
   private $connection;

   public function __construct($db)
   {
      $this->connection = $db;
   }

   public function getData($table, $userId, $period = '')
   {
      return LoadDataFromDB($this->connection, $table, $userId, $period);
   }

   public function getLastId($table)
   {
      return GetLastId($this->connection, $table);
   }

   public function deleteData($table, $id)
   {
      $this->connection->query("DELETE FROM `$table` WHERE `id` = $id")->fetchAll(PDO::FETCH_ASSOC);
      return 'ok';
   }

   public function addData($table, $userId, $options, $time = '')
   {
      $values = "";

      switch ($table) {
         case 'users':
            $checkByLogin = $this->connection->query("SELECT * FROM `users` WHERE `login`= '$options[1]'")->fetchAll(PDO::FETCH_ASSOC);
            $checkByEmail = $this->connection->query("SELECT * FROM `users` WHERE `email`= '$options[0]'")->fetchAll(PDO::FETCH_ASSOC);

            if (count($checkByLogin) > 0 || count($checkByEmail) > 0)
               return 'double';

            $columns = "`email`, `login`, `password`";
            $options[2] = hashPassword($options[2]);
            break;

         case 'accounts':
            $columns = "`name`, `smtp`, `email`, `password`";
            break;

         case 'contacts':
            if (count($this->connection->query("SELECT * FROM `contacts` WHERE `email`= '$options[1]'")->fetchAll(PDO::FETCH_ASSOC)) > 0) {
```

```
                return 'double';
            }

            $columns = "`name`, `email`, `time`";
            array_push($options, date('d.m.y H:i'));
            break;

        case 'messages':
            $columns = "`message`, `recipientEmail`, `time`";
            array_push($options, date('d.m.y H:i'));
            break;

        case 'tasks':
            $columns = "`task`, `time`";
            $options = [json_encode($options)];
            array_push($options, $time);
            break;
    }

    if (!empty($userId)) {
        $columns = $columns . ', `userId`';
        array_push($options, $userId);
    }

    foreach ($options as $option) {
        if (!empty($values)) {
            $values = "$values, '$option'";
        } else {
            $values = "'$option'";
        }
    }

    $this->connection->query("INSERT INTO `$table`($columns) VALUES ($values)")->fetchAll(PDO::FETCH_ASSOC);

    if ($table === 'users') {
        $mail = new Email();
        $result = '';
        $id = $this->connection->query("SELECT * FROM `users` WHERE `email`= '$options[0]'")->fetchAll(PDO::FETCH_ASSOC)[0]['id'];
        $mail->sendMail(
            '',
            (object) [
                'senderEmail' => 'konstantinbylbas@gmail.com',
                'senderPassword' => 'peoc orhi xgyu lddx',
                'smtp' => 'smtp.gmail.com',
                'recipientEmail' => $options[0],
                'id' => $id
            ],
            'register',
            $result
        );
    }

    return 'added';
}
```

```
        public function editData($table, $id, $column, $value)
        {
            if ($column === 'password')
                $value = hashPassword($value);

            $this->connection->query("UPDATE `$table` SET `$column` = '$value' WHERE `id` = $id")->fetchAll(PDO::FETCH_ASSOC);
            return 'success';
        }

        public function restorePassword($options)
        {
            try {
                $mail = new Email();
                $email = $options[0];
                $result = '';
                $user = $this->connection->query("SELECT * FROM `users` WHERE `email`= '$email'")->fetchAll(PDO::FETCH_ASSOC);
                if (!count($user)) {
                    return 'no user with such email';
                }

                $password = $this->generateRandomPassword();
                $result = $mail->sendMail(
                    '',
                    (object) [
                        'senderEmail' => 'konstantinbylbas@gmail.com',
                        'senderPassword' => 'peoc orhi xgyu lddx',
                        'smtp' => 'smtp.gmail.com',
                        'recipientEmail' => $email,
                        'newPassword' => $password
                    ],
                    'restore'
                );

                if ($result === 'Message sent') {
                    $this->editData('users', $user[0]['id'], 'password', $password);
                    return 'restored';
                } else return 'try later';
            } catch (\Throwable $th) {
                return 'error: ' . $th;
            }
        }

        private function generateRandomPassword($length = 10)
        {
            $characters = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
            $password = '';

            for ($i = 0; $i < $length; $i++) {
                $index = rand(0, strlen($characters) - 1);
                $password .= $characters[$index];
            }

            return $password;
        }
    }
```

objects/email.php

```
<?php

include_once 'types/emailTemplates.php';

use PHPMailer\PHPMailer\PHPMailer;

class Email {

   private $database;
   private $db;
   private $data;

   public function __construct() {
      $this->database = new Database();
      $this->db = $this->database->getConnection();
      $this->data = new Data($this->db);
   }

   public static function sendMail($userId, $emailData, $type) {
      $emailInstance = new self();
      $mail = new PHPMailer(true);
      $lastMessage = $emailInstance->data->GetLastId('messages')[0]['id'] ?? 0;
      $body = self::getBody($type, $emailInstance->data, $emailData, $lastMessage);

      try {
         $mail->IsSMTP();
         $mail->CharSet = 'UTF-8';

         $mail->Host       = $emailData->smtp;
         $mail->SMTPAuth   = true;
         $mail->Username   = $emailData->senderEmail;
         $mail->Password   = $emailData->senderPassword;
         $mail->SMTPSecure = PHPMailer::ENCRYPTION_SMTPS;
         $mail->Port       = 465;

         $mail->setFrom($emailData->senderEmail, '');
         $mail->addAddress($emailData->recipientEmail);

         $mail->isHTML(true);
         $mail->Subject = ' ';
         $mail->Body    = $body;
         $mail->AltBody = '';

         $mail->SMTPDebug = 0;
         $mail->send();

         if ($userId) {
            $emailInstance->data->addData('messages', $userId, [$body, $emailData->recipientEmail]);
         }

         return 'Message sent';
      } catch (Exception $ex) {
         return "Message could not be sent. Mailer error: {$mail->ErrorInfo}";
      }
   }
```

```
    public static function getBody($type, $data, $emailData, $lastMessage = 0) {
        $message = '';

        switch ($type) {
            case 'restore':
                $message = getRestoreTemplate($emailData->newPassword);
                break;

            case 'register':
                $message = getRegisterTemplate($emailData->id);
                break;

            default:
                $users = $data->getData('users', '');
                foreach ($users as $user) {
                    if ($user['id'] === $emailData->senderId) {
                        $message = getDefaultTemplate($user['messageTemplate'], $lastMessage + 1);
                        break;
                    }
                }
                break;
        }

        return $message;
    }

}
```

objects/login.php

```
<?php

class Login
{

    private $connection;
    private $dataFromDB;

    public function __construct($db)
    {
        $this->connection = $db;
        $this->dataFromDB = LoadDataFromDB($this->connection, 'users');
    }

    public function checkUser($options)
    {
        foreach ($this->dataFromDB as $user) {
            if ($user['login'] === $options[1] && $user['password'] === hashPassword($options[2])) {
                return array(
                    'id' => $user['id'],
                    'login' => $user['login'],
                    'messageTemplate' => $user['messageTemplate'],
                    'token' => $this->getToken($user['login'], $user['password'])[0]
                );
            }
        }
        return 'no such user';
```

```
    }

    public function checkToken($token, $login)
    {
        $flag = 0;

        foreach ($this->dataFromDB as $user)
            if ($user['login'] === $login)
                for ($i = 0; $i < 5; $i++)
                    if ($token === $this->getToken($login, $user['password'])[$i])
                        $flag = 1;

        if ($flag === 1)
            return array(
                'id' => $user['id'],
                'login' => $user['login'],
                'messageTemplate' => $user['messageTemplate'],
                'status' => 'ok'
            );
        else return 'invalid token';
    }

    private function getToken($login, $password)
    {
        $tmp = [];

        for ($i = 0; $i < 5; $i++)
            array_push($tmp, md5('mad' . $password . date('d.m.y', strtotime(date('d.m.y') . " -$i  day")) . 'fox' .
$login . 'mailer'));
        return $tmp;
    }
}
```

services/hashPassword.php

```
<?php

function hashPassword($password)
{
    return md5('mad' . md5($password) . 'fox' . '2022');
}
```

services/loadDataFromDB.php

```
<?php

function LoadDataFromDB($db, $table, $userId = '', $period = '')
{
    $where = '1';

    if (!empty($userId)) {
        $userId = "`userId` = '$userId'";
    }

    if (!empty($time)) {
        $where = $time;

        if (!empty($userId)) {
            $where .= ' AND ' . $userId;
```

```
        }
    } else if (!empty($userId)) {
        $where = $userId;
    }

    $data = $db->query("SELECT * FROM $table WHERE " . $where)->fetchAll(PDO::FETCH_ASSOC);
    if (count($data)) {
        return $data;
    } else {
        return 'empty';
    }
}

function GetLastId($db, $table)
{
    return $db->query("SELECT MAX(id) as id FROM `" . $table . "`")->fetchAll(PDO::FETCH_ASSOC);
}
```

services/toProcessTasks.php

```
<?php

function toProcessTasks()
{
    $database = new Database();
    $db = $database->getConnection();
    $data = new Data($db);
    $tasks = $data->getData('tasks', '');
    $mail = new Email();

    if (!empty($tasks) && gettype($tasks) === 'array')
        foreach ($tasks as $task) {
            $emailData = json_decode($task['task']);
            if (strpos($emailData->recipientName, "u2019"))
                $emailData->recipientName = str_replace("u2019", "'", $emailData->recipientName);
            $mail->sendMail($emailData, 'userTemplate');

            $data->deleteData('tasks', $task['id']);
        }
}
```

types/emailTemplates.php

```
<?php

function getRestoreTemplate($newPassword) {
    return '
<h2>Вітаємо!</h2>
<br/>
<p>Дякуємо за звернення та співпрацю. Ваш новий пароль <b>' . $newPassword . '</b></p>
<br/>
<p>Гарного дня</p>';
}

function getRegisterTemplate($id) {
    return '
<h2>Вітаємо новий користувачу!</h2>
<br/>
```

```
        <p>Дякуємо за рєстрацію! Для підтвердження пошти та завершення реєстрації <a
href="http://localhost/mailer_api/?verifyEmail=' . $id . '">натисни сюди</a></p>
        ';
        }

        function getDefaultTemplate($messageTemplate, $recipientId) {
          return $messageTemplate . '<img style="width:1px;height:1px;"
src="http://localhost/mailer_api/?mailOpened=' . $recipientId . '" />';
        }
```