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

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Олійника Данііла Валерійовича*
(ПІБ)

академічної групи *122-17-3*
(шифр)

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

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

на тему: «Розробка веб-додатку для замовлення доставки продуктів споживачам»

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи				
розділів:				
спеціальний				
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро, 2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

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

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

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-17-3* *Олійника Данііла Валерійовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи **«Розробка веб-додатку для замовлення доставки продуктів споживачам»**

затверджена наказом ректора НТУ «ДП» від *07.06.2021 р.* № *317-с*

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

Завдання видав _____ _____
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Олійник Д.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

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

РЕФЕРАТ

Пояснювальна записка: 80 с., 18 рис., 0 табл., 3 додатків, 22 джерел.

Об'єкт розробки: інформаційна система для доставки продуктів споживачам.

Мета дипломного проекту: мета роботи полягає у «Розробка веб-додатку для замовлення доставки продуктів споживачам».

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

Перший розділ присвячено аналізу предметної області, визначенню актуальності завдання та призначення інформаційної системи, розроблена постановка завдання, технологій та програмних засобів.

Другий розділ включає у себе виконання аналізу існуючих рішень, обрання платформи для розробки, проектування та розробку програми, наведення опис алгоритму і структури функціонування програми, наведення характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення. Практичне значення полягає у створенні додатку, що допоможе споживачам та власникам магазинів без перешкод сортувати та доставляти продукти до зазначенного споживачем місця.

Актуальність даного програмного продукту визначається скороченням часу, який витрачається споживачам на покупку необхідних для них товарів. Також це є досить актуальною темою на сьогоднішній час через карантинний режим.

Список ключових слів: JAVA SCRIPT, HYPERTEXT MARKUP LANGUAGE, VISUAL STUDIO CODE.

ABSTRACT

Explanatory note: 80 pp., 18 figs., 0 tabl., 3 appendices, 22 sources.

Object of development: information system for delivery of products to consumers.

The purpose of the diploma project: the purpose of the work is to " Development of a web application for ordering product delivery to consumers."

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

The first section is devoted to the analysis of the subject area, determining the relevance of the task and the purpose of the information system, developed the statement of the task, technology and software.

The second section includes analysis of existing solutions, selection of a platform for development, design and development of the program, description of the algorithm and structure of the program, characteristics of the parameters of technical means, describes the call and download of the program, describes the program.

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

The practical significance is to create an application that will help consumers and store owners to easily sort and deliver products to the place specified by the consumer.

The relevance of this software product is determined by the reduction of time spent on consumers to purchase the goods they need. It is also a very relevant topic today due to the quarantine regime.

Keyword list: JAVA SCRIPT, HYPERTEXT MARKUP LANGUAGE, VISUAL STUDIO CODE.

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

СУБД – СИСТЕМА УПРАВЛІННЯ БАЗИ ДАНИХ
PC – PERSONAL COMPUTER
JS – JAVA SCRIPT
ТП – ТЕХНІЧНА ПІДТРИМКА
ПЗ – ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ
VSC – VISUAL STUDIO CODE
PHP - HYPERTEXT PREPROCESSOR
SQL - SCRUCTURED QUERY LANGUAGE
HTML - HYPERTEXT MARKUP LANGUAGE
CSS - CASCADING STYLE SHEETS

ЗМІСТ

ВСТУП

Завдання даного дипломного проекту та об'єкт його діяльності - інформаційна автоматична система для доставки продуктів споживачам, безпосередньо пов'язані з напрямом підготовки «Комп’ютерні науки» та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених виробничих функцій, типових задач діяльності, умінню та компетенціям, якими повинні володіти бакалаври напряму 122 «Комп’ютерні науки».

Розроблена інформаційна система буде призначена для використання у сфері доставки їжі споживачам. Інформаційна система – це сукупність технічних впорядкованих між собою засобів, які використовуються для збереження та обробки інформації безпосередньо самим користувачем, тому це дуже важлива складова частина усіх додатків.

Актуальність цього додатку особливо можна відчувати внаслідок пандемії, яка виникла у всьому світі, але також треба зазначити, що додатки з доставки їжі були актуальні з першого дня появи. Служба доставки їжі це досить прибутковий бізнес, який, наприклад, на відміну від масштабних проектів ресторанів не вимагає площі або обладнання, тому є більш зручним рішенням для сучасної людини, яка хоче себе захистити та зберегти свій час.

Також треба зазначити, що використовуючи додаток або сайт для замовлення їжі, ресторани можуть підвищити коєфіцієнт лояльності споживачів через те, що замовник обирає товари, знаходячись у комфортній обставноці і йому не потрібно контактувати з персоналом, який знаходиться, наприклад, у ресторані або магазинах. Треба визначити, що перспектива розвитку таких додатків є дуже переважливою у наш час, адже споживачам нічого не потрібно робити.

Можна проаналізувати дохід сайтів або додатків за останній час і ми побачимо на статистиці, що за дуже короткий проміжок часу, додатки стали не

тільки актуальним для людей, але й підняли рівень економіки.

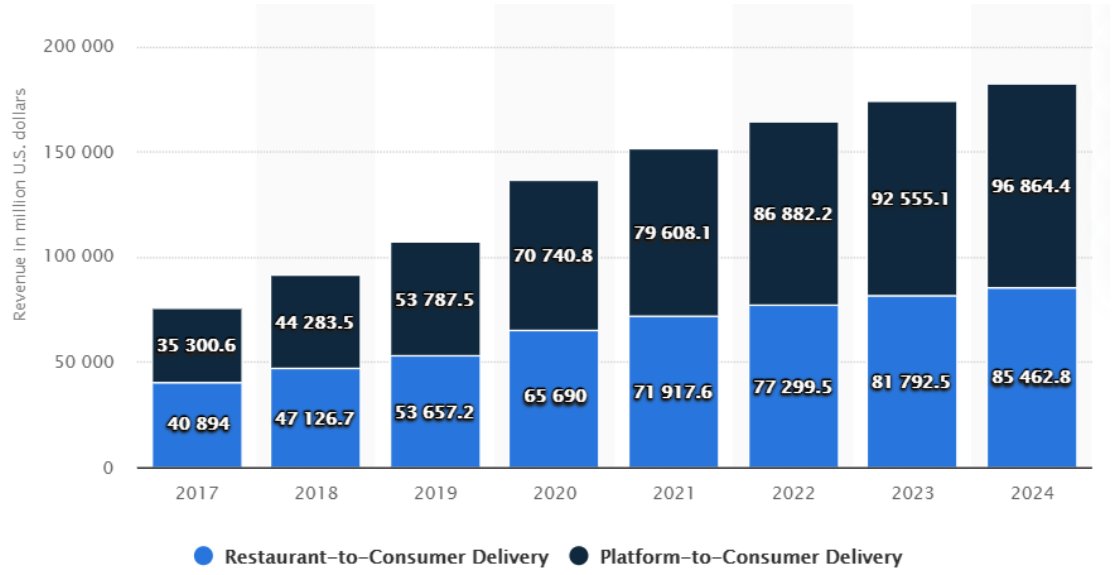


Рис. 1.1. Рівень профіту з додатків на сьогоднішній день

Для виконання зазначеного рівня, постійно проводяться роботи з покращенням сайту або додатку для ресторанів і самостійних компаній. Також є дуже примітним те, що у графіку окремо відображається прибуток платформи з доставки їжі (маються на увазі такі крупні компанії, як Glovo, RS, Raketa) та ресторанів. Але у будь-якому випадку, ресторанам не зовсім вигідно робити свій власний додаток, тому що це більш ресурсовитратно, ніж зробити сайт; також ресторани досить часто замовляють послуги у деяких платформ, тому може на перший погляд здаватися, що платформи та ресторани мають приблизно однаковий дохід, але це не зовсім коректно. Ці висновки були засновані на безперервному документуванні прибутку ресторанів та платформ, які доставляють їжу.

Тому проблема, розглянута в цьому дипломному проекті, є актуальною та має широке практичне призначення як на сьогоднішній день, так і в майбутньому.

Метою даного проекту є розробка системи постачання їжі для споживачів.

Для досягнення поставленої мети необхідно вирішити основні завдання:

- аналіз платформ та сайти ресторанів, які доставляють їжу споживачам;

- уточнення вимог до написання інформаційної системи, яка дозволить без перешкод користуватися сайтом, який буде розподіляти та відправляти їжу користувачам;

- розробка алгоритму, інтерфейсу, бази даних і реалізації програми.

У відповідності до проведеного аналізу поставлені основні функціональні задачі перед системою:

- прискорення розвитку більш інноваційних систем для споживачів;

- зниження витраченого користувачем часу на покупку обраних товарів;

- збільшення ефективності ведення обліку несправностей;

В цілому, надана інформаційна система забезпечує та підвищує рівень задоволення клієнтів та значно полегшує ресторанний бізнес через можливість доставляти їжу на будь-яку відстань та у будь-який проміжок часу. Розроблену інформаційну систему можуть використовувати усі бажаючі споживачі, які хочуть полегшити життя та економити час.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ І ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

З упевненістю можна заявити, що дана сфера діяльності людей стала невід'ємною частиною їхнього життя. Велика частина ресторанів створила свої власні сайти з доставки їжі, а також керує окремою групою людей, яка технічно підтримує сайт і доставляє їжу замовникам. Також, не менш важливим пунктом є реакція ринку на глобальну пандемію, так як це питання актуальне в усьому світі вже другий рік. Домінантними представниками на ринку послуг онлайн-доставки їжі є Uber eats (Zomato), takeaway.com, Swiggy, Deliveroo, Amazon.

Наразі глобальний ринок неможливо собі уявити без сучасних технологій, які полегшують нам життя. На жаль, у нас в Україні онлайн-послуги тільки почали завойовувати свою популярність. Мировий ринок послуг з доставки їжі, що розглядається в цьому дипломі, розділений за типами на платформа (додаток) - постачання клієнту, ресторан – постачання клієнту. За типом розподілу на класи, доставка може бути відображена у вигляді веб-сайту, у вигляді мобільного додатку та за способом оплати, наприклад: накладений платіж, онлайн-платіж, готівковий платіж.

Також, якщо судити з точки зору економіки, то вартість доставок страв й логістики, тобто кур’єри, буде основним стримуючим фактором розвитку для ринку онлайн-послуг з доставки їжі, але ні в якому разі не з технічної причини. Вартість повина включати витрати на виконання замовлення кур’єром, вартість доставки, адаптацію бізнес-ресурсів до динамічного ринкового попиту. Крім того, є витрати на картонні коробки для пакування страв, на бензин, кілометраж й вартість найму водія або кур’єра – це є головною перешкодою створення

сайту з розподілу їжі. Ланцюжок поставок і логістика повинні бути налагоджені, щоб уникнути псування продуктів з обмеженим терміном зберігання. Користувачі смартфонів є основними замовниками в індустрії з онлайн-доставки їжі, тому має сенс робити не сайт, а платформу або додаток на якому користувачі зможуть спокійно обирати товари, що їм до вподоби. Але як вже було раніше сказано, сайти є більш функціонуючим ланцюжком у сфері доставки їжі, ніж додатки, тобто в них є більше переваг. Також, збільшення числа користувачів телефонів відображає збільшення кількості покупок в Інтернеті продуктів харчування та напоїв. Глобальна кількість підписників електронної комерції досягла вже 1,5 мільярда на 2019 рік та судячи зі статистики, буде зростати ще на 800 мільйонів кожен рік. Отже, зростання кількості користувачів гаджетів і проникнення їх до Інтернету стимулює зростання онлайн-сервісів доставки їжі.

Однак, перш ніж розробляти платформу чи сайт з доставки їжі, важливо зрозуміти, який саме алгоритм дій повинен працювати. Можна взяти на прикладі будь-який додаток з доставки їжі, проаналізувати його та почати власне виробництво. По-перше, ми бачимо інтерфейс, а тільки потім аналізуємо об’єм роботи, яку потрібно буде виконати.

Такі програми або сайти, що приймають замовлення на їжу, обробляють замовлення та доставляють їжу. Додаток надає користувачам безліч функцій, щоб зробити процеси замовлення та доставки максимально зручними для всіх. Деякі з примітних розповсюджених функцій включають в себе наступне:

- Рекомендації страв від ресторанів для споживачів.
- Фільтри пошуку товарів по обраним смакам користувача (наприклад, вегетаріанська їжа, рослинне молоко, замінювач цукру тощо).
- Налаштовуємі деталі доставки (мається на увазі готівкова плата за замовлення або безготівкова, тобто карткою у додатку чи на сайті).

- Відстеження замовлення є також досить важливим пунктом, тому що більшість користувачів нервуються через неможливість дізнатися час прибуття замовлення.
- Яскравий та інтересний інтерфейс також є невід’ємною частиною майбутнього сайту або додатку.
- Зручний функціонал. Цей пункт є основним, тому що нагромадження функцій може знижувати ефективність функціоналу вашого сайту або додатку. Мінімальна кількість взаємодій.

Також, якщо порівнювати платформу з сайтом, то сайт є більш вигідним аналогом додатку. Додаток є досить затратним не тільки з точки зору фінансів та піар кампаній але й затраченого часу на його написання та розповсюдження. Сайт є більш зручним не тільки для користувачів але й для власників ресторанів, тому що вони мають сплачувати лише логістичні послуги, тобто перевозку товарів.

У деяких аспектах доставки їжі можна сказати, що технічні протиріччя є мінімальним для сайтів та додатків. Технічні протиріччя – це така ситуація, при якій спроба покращити одну з характеристик об'єкта, погіршується інша його характеристика або показник. Між цими показниками існує співвідношення, яке й називають технічним протиріччям. Але у сайтів для доставки їжі ці протиріччя мінімальні, через простий дизайн та стислий функціонал.

1.2. Призначення розробки та галузь застосування

Призначення розробки та галузь застосування – це застосування інформаційної системи для доставки продуктів споживачам. На разі, така СУБД є досить актуальною у наш час та може використовуватись як з PC, так і з ваших смартфонів. Ця система застосовується для сортування, розподілення та час доставки продуктів. По-перше, будь-який користувач сайту або платформи

буде спроможний використати таку систему з будь-якої точки країни, або, навіть світу. По-друге, для використання такої системи, не потрібно мати потужний гаджет через те, що функціонал дуже мінімальний. Система з розподілення їжі може використовуватись не тільки серед звичайних користувачів, але й мати спеціальне призначення. Наприклад, такі системи можуть використовуватись в армії для розподілення їжі між солдатами серед невеликих груп людей. Також таку систему можна використовувати для більш складних задач, наприклад для розподілення їжі в університеті або великому офісі.

1.3. Підстава для розробки

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

Отже підставами для розробки (виконання кваліфікаційної роботи) є:

- ОКХ та ОПП за напрямом підготовки «Комп’ютерні науки»;
- Навчальний план та графік навчального процесу;
- Наказ ректора Національного технічного університету «Дніпровська політехніка» №317-с від 07.06.2021 року;
- Завдання на дипломний проект на тему: «Розробка інформаційної системи для доставки продуктів споживачам».

1.4. Постановка завдання

Метою даної кваліфікаційної роботи є створення інформаційної системи для доставки їжі.

Система для доставки їжі повинна реалізувати наступні показники:

- Мета розробки такої системи полягає в поліпшенні та полегшенні людського життя за допомогою доставки їжі. Дана інформаційна система може бути застосована у різних контингентах суспільства. Наприклад, у навчальному закладі, для масових заходів, торговельних центрах, медичних та реабілітаційних центрах тощо. Основне завдання полягає в написанні функціонального простого сайту на мові програмування JS.
- Інформаційна система для доставки їжі може бути використана як зі сторони незалежних розробників, які не мають власної компанії, також зі сторони таких закладів, як ресторани. Інформаційна система буде розроблена і призначена спеціально як для замовників, так і для користувачів. Однак, якщо ресторани не мають власного сайту, вони можуть скористатися послугами таких платформ за додаткову комісію або платню.
- По-перше, сайт повинен складатися з яскравого та зрозумілого інтерфейсу, перед користувачем повинні бути основні функції сайту на головній сторінці. Також, важливе значення має коректне відображення інтерфейсу, як на телефоні, так і на PC. По-друге, можна відстежувати стан додатку або сайту судячи з прийманих замовлень.
- Чіткий правильний структурований код, який дозволяє приймати інформацію від користувача, тобто замовлення, обробляти цю інформацію, тобто записувати замовлення та відсилати до ресторанів або продуктових магазинів. Обробка замовлень повинна бути у порядку живої черги, тобто обробляти замовлення від одного користувача до останнього, але робити це без жодних затримок, тобто автоматично коригувати неправильні замовлення. Також є важливим факт контролю замовлень, які надходять до сайту або платформи.
- У разі, якщо код був написаний структуровано невірно, припиняється автоматична обробка замовлень на сайті або платформі. Сайт є більш гнучким матеріалом, тому зафіксувати та виправити помилки легше, ніж

зробити це на платформі. Тому, у такому разі, деякі з платформ мають навіть власну технічну підтримку, яка буде обробляти запити від користувачів, які мають індивідуальні збії у програмі.

- Така інформаційна система є досить гнучкою, тому вона може бути пов’язана наприклад з такими задачами, як розподіл замовлень індивідуально кожному з кур’єрів, які знаходяться недалеко від точки доставки замовлення. Також, можуть бути кур’єри, які окремо займаються замовленнями з ресторанів чи продуктових магазинів тощо.
- Як вже було сказано раніше, система може розподіляти між собою замовлення, які йдуть окремим кур’єрам. Також, треба додати функцію написання листа до ТП, такі листи будуть автоматично сортуватися по темі написання та надходити до відповідних відділів, які займаються конкретними справами. Також є досить важливою функція, яка буде відстежувати на Google Maps кур’єра, який повинен відвезти замовлення до користувача. Така функція буде спеціально додана для спокійного та комфортного користування, щоб чітко відстежувати час доставки замовлення.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Вимоги до сайту полягають в обробці, розподіленню, контролю інформації, яка надходить до нього від користувачів. Головне меню користувача або замовника повинно складатися лише з корисних та діючих опцій, що відповідають ходу їх процесу. У них повинні бути варіанти наступні варіанти:

- Реєстрація на сайті або в додатку.
- Список ресторанів або продуктових магазинів, які також можуть займатись доставкою. Також повинні бути фільтри, які допомагають користувачеві здійснити вибір продуктів, які йому до смаку.

- Декілька варіантів платежів, тобто готівковий платіж або безготівковий.
- Пропозиції та компейн промокоди.
- Відстеження в реальному часі і відомості про відповідальне за доставку.

1.5.2 Вимоги до інформаційної безпеки

Для захисту ПЗ в першу чергу необхідно обмежити доступ до коду усіх, окрім системних адміністраторів та технічної підтримки, яка також працює над виправленням помилок. Також, у кожного користувача сайту або додатку буде свій власний унікальний ідентифікаційний код, який допоможе захистити облікові записи.

1.5.3 Вимоги до складу та параметрів технічних засобів

Почати треба с того, що сайту може працювати на будь-якій операційній системі, тобто:

- Windows 10
- Windows 7
- Windows 8
- Linux
- Unix

Для коректного відображення сайту потрібно мати один з перечислених браузерів:

- Internet Explorer 8.0.0
- Opera 5.0
- Google Chrome 10.0
- Mozilla Firefox 13.0

Мінімальні системні вимоги:

- Процесор класу Intel Core
- Не менше 2 ГБ ОЗУ
- Вільного місця на жорсткому диску приблизно від 100 МБ.

Вимоги до обладнання:

- Монітор
- Миша
- Навушники
- Працюючий PC або ноутбук.

1.5.4 Вимоги до інформаційної та програмної сумісності

Технічні вимоги до комп'ютера, який відповідає мінімальним системним вимогам для розробки і розгортання створюваної програми, інсталяції інструментальних засобів розробки, що враховує особливості експлуатації програмного забезпечення, приведені для всіх використовуваних програмних засобів. У такому випадку для розробки сайту буде використовуватись Visual Studio Code. Також для розробки інформаційної системи будуть використанні такі мови програмування:

- Java Script
- PHP
- SQL
- HTML
- CSS

Також, написання коду може бути на будь-якій операційній системі, наприклад:

- Windows 10
- Windows 7
- Windows 8
- Linux

- Unix

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Функціональне призначення програми складається з автоматизації інформаційної системи для доставки їжі користувачам. Функціонал даної програми хоч і обмежений, однак, такі функції як вхід в обліковий запис адміністратора, клієнта, кур'єра і реєстрація разом з збереженнями даних, передбачена.

- Обліковий запис адміністратора дозволяє нам додавати потрібний товар, вивантажувати картинки, опис до товару, кількість одиниць товарів на складі, змінювати написи і оформлення сторінки і сайти в режимі реального часу.

- Обліковий запис кур'єра дозволяє додавати замовлення, які потрібно виконати, в свою власну чергу, щоб інші кур'єри не змогли повторно взяти той же замовлення, також профіль кур'єра має додаткову панель на якій можна відстежувати виконані або поточні замовлення. Також кур'єр може переглядати всі свої виконані замовлення за останній час.

- Обліковий запис клієнта дозволяє заходити в додаток і переглядати вивантажені товари на сайті, вибирати потрібний товар виходячи з категорій, також додатково можна вибирати кількість товарів, яке клієнт хоче замовити, а також очікувати доставку товарів від кур'єра.

Функціонал сайту і платформи достатньо простий, але ефективний, так як в роботі він ідеально підійде з точки зору замовника, так як є функції, які дозволяють редагувати сайт в теперішньому часі, так і з точки зору споживача, маючи можливість купувати товари у великій кількості. Дана платформа забезпечує необхідними ресурсами не тільки замовника і споживача, але і виконавців теж, тобто кур'єрів.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної області інформаційної системи, що проектується, не передбачають застосування математичних методів, при розробці інформаційної системи з доставки їжі математичні методи не використовувались.

2.3. Опис використаних технологій та мов програмування

Для створення структури всієї програми використовувалося інтегроване сере довище розробки програмного забезпечення Microsoft Visual Studio, а також такі мови програмування як: JS, PHP, SQL, HTML і CSS. Також нам варто позначити кожну мова програмування окремо:

- JS (JavaScript) - це мова програмування, який використовується в першу чергу для більш складних цілей в документі HTML. JS - це мова, яка дозволяє змінювати інтерфейси веб додатки, додавати або змінювати різні функціональні можливості сайту. Сторонні API можуть дозволити нам додавати різні функції, які вже існують на інших сайтів. Також ви можете застосувати до вашого HTML документу сторонні фреймворки і бібліотеки, що дозволить вам прискорити створення сайтів і додатків.

- PHP (Hypertext Preprocessor) - це мова програмування, який потрібен для статистичного відображення сторінки, а також для "довголіття" вашого сайту. Тобто, дана мова використовується для щоденного оновлення сайту, надходження нової свіжої інформації, яка буде актуальна кожен день для користувачів. Дана мова програмування використовується для написання "сценаріїв", які завантажуються в HTML документ, щоб відображати нестатичність картинки. Справа в тому, що документ HTML здатний відображати тільки статичну інформацію на сайті, а PHP показує "свіжу".

- SQL (Structured Query Language) - це мова програмування, яка може використовуватися як ефективний спосіб для таких функцій, як: збереження в БД, пошуку їх частин, поновлення, вилучення з бази даних і видалення звідти. Взаємодія з БД відбувається швидко навіть в ситуаціях, коли обсяги даних при завантаженні досить великі. SQL – це досить зручна мова для великої кількості інформації, так як дозволяє миттєво розробнику перерозподіляти інформацію.

- HTML (HyperText Markup Language) - це мова програмування, яка дозволяє розробнику створювати і структурувати безліч розділів, заголовків, посилань і цілі блоки для веб-сторінок і додатків. Це мова, яка дозволяє створювати структуру web-сторінок.

- CSS (Cascading Style Sheets) - це скоріше не мова програмування, а правила оформлення HTML документа. CSS дозволяє розробнику встановлювати розмітку сторінки, розмір шрифтів, колір, заголовки, а також змінювати шрифт. Можна сказати, що CSS безпосередньо

відповідає за візуальну складову сайту, так як жоден HTML документ не може обійтися без CSS.

2.4. Опис структури системи та алгоритмів її функціонування

А) Алгоритм заданої системи полягає в тому, що кожен з користувачів взаємопов'язаний один з одним. Наприклад, адміністратор завантажує на сервер їжу або страви з певним обмеженою кількістю, поміщаючи в окрему групу продуктів. Користувач бачить, що новий продукт був доданий в обмеженою кількості і виходячи зі своїх потреб, замовляє продукт з тієї чи іншої категорії. Тобто користувачем здійснюється купівля продукту, далі дане замовлення формується і надходить в базу даних кур'єрів, між ними дані замовлення розподіляються. Тобто, кожен кур'єр вибирає собі замовлення виходячи з адреси та методу оплати.

Також дана система спрямована на те, щоб алгоритм виконувався у виключній послідовності, тобто так, щоб не виникало збоїв, які заважають циклу повторення алгоритму.

В) Структура системи складається з сполучних компонентів, які виражені у вигляді облікових записів. У даній інформаційній системі кожен користувач нерозривно пов'язаний з іншим. Наприклад, кур'єр пов'язаний з клієнтом, а клієнт з кур'єром і адміністратором, в свою чергу адміністратор пов'язаний з клієнтом і кур'єром. Таким чином дані користувачі утворюють цикл, тобто періодичність повторюваних дій. Клієнт замовляє товар, замовлення утворюється на сайті і приймається кур'єром, адміністратор відповідає за поставку необхідних товарів і за коригування на сайті.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Дані користувача збираються в формі у користувача, відправляються на сервер і далі обробляються (незалежно від типу користувача); також по заздалегідь створеному шаблоном виводяться всі товари і користувач може їх вибрати в потрібній кількості і замовити, під час вибору товару відбувається запис в базу даних, тобто всі товари, які були обрані користувачем, не будуть загублені, навіть при зміні пристрою; потім створюється замовлення, який передається вільному кур'єру.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

Для написання даної інформаційної системи використовувався персональний комп'ютер, мишка, клавіатура і гарнітура.

2.6.2. Використані програмні засоби

Створення даної інформаційної системи було створено на операційній системі Windows 10.

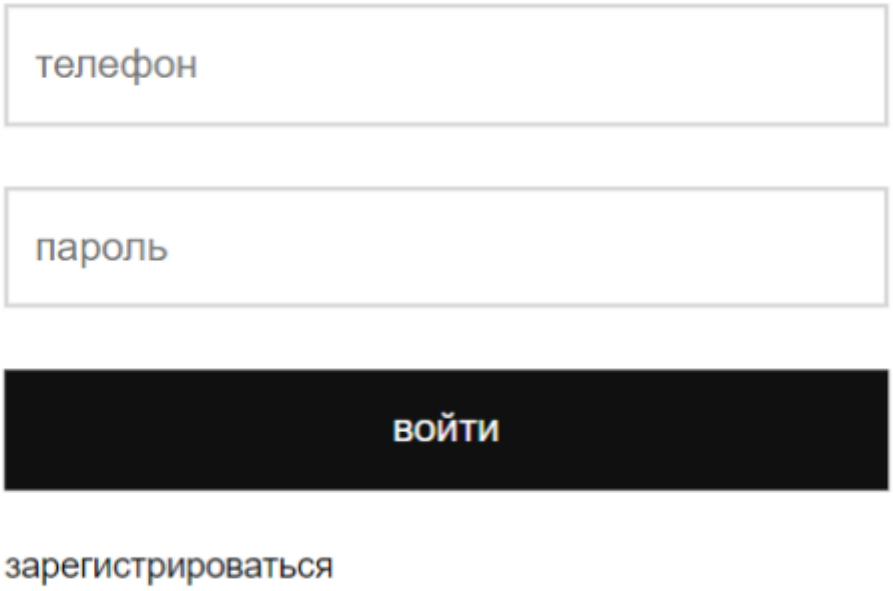
Також для розробки даної інформаційної системи використовувалася платформа Visual Studio Code.

2.6.3. Виклик та завантаження програми

Програма з розподілу і доставки їжі може бути викликана як через накопичувач, тобто флешку або диск, так і через перехід по посиланню. Тобто, поширення посилання відбувається безпосередньо через браузер, який використовується користувачами. Дана програма також може бути завантажена на платформу через яку буде виконуватися вхід в мобільний додаток через яке можна буде здійснити вхід. Також програму можна завантажити з браузера Google Chrome, так як була додана подібна опція.

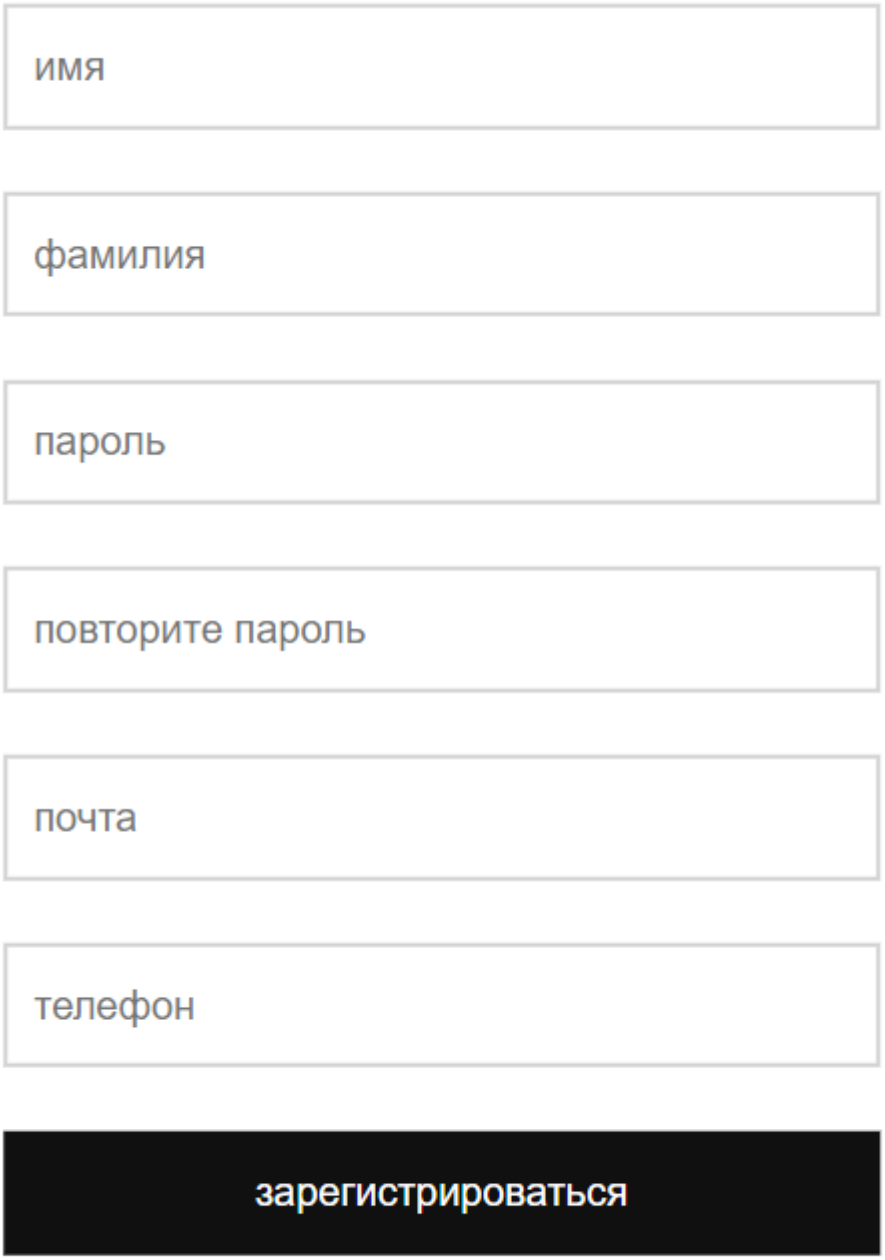
2.6.4. Опис інтерфейсу користувача

Перше, що бачить користувач, коли заходить на веб сайт - це панель через яку можна увійти в свою обліковий запис або зареєструватися на сайті:



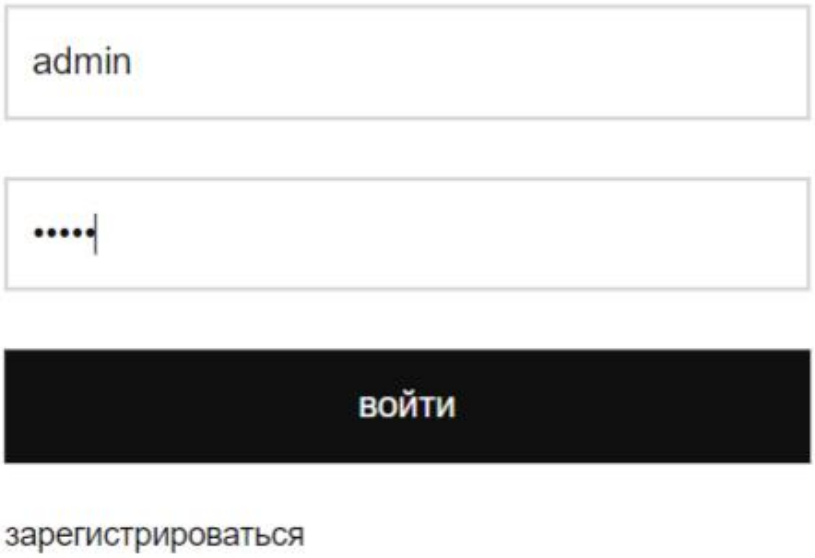
1.2. Початкове меню

У вкладці реєстрації нам доступні стандартні дані користувача:

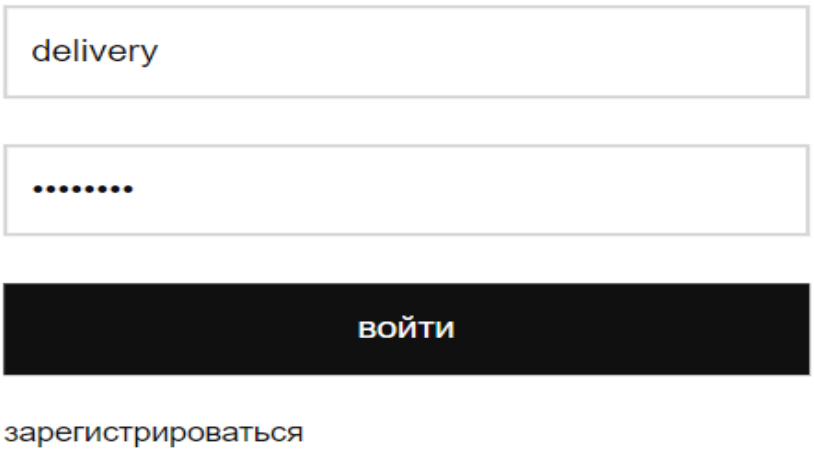


1.3. Меню реєстрації

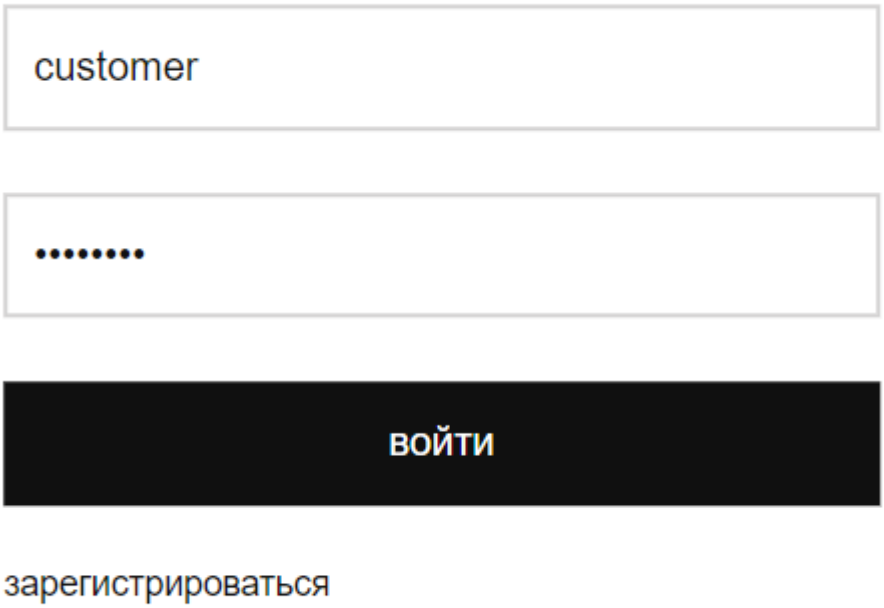
Також користувачам доступні певні функції в залежності від їх акаунтів:



1.4. Акаунт адміністратора



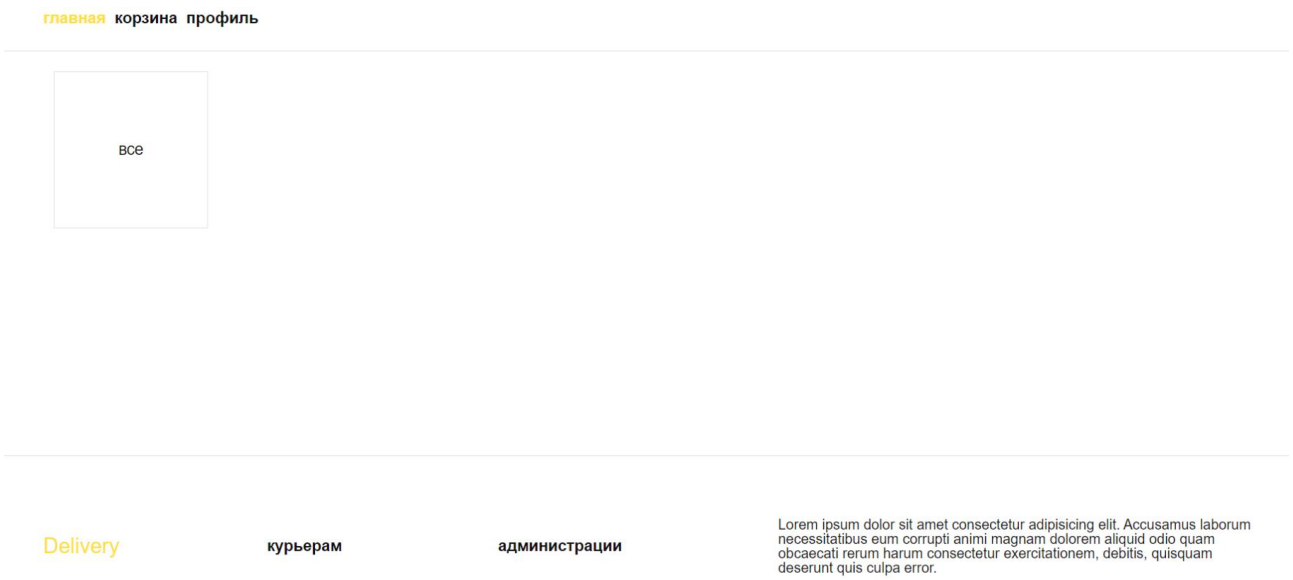
1.5. Акаунт кур’єра



1.6. Акаунт покупця

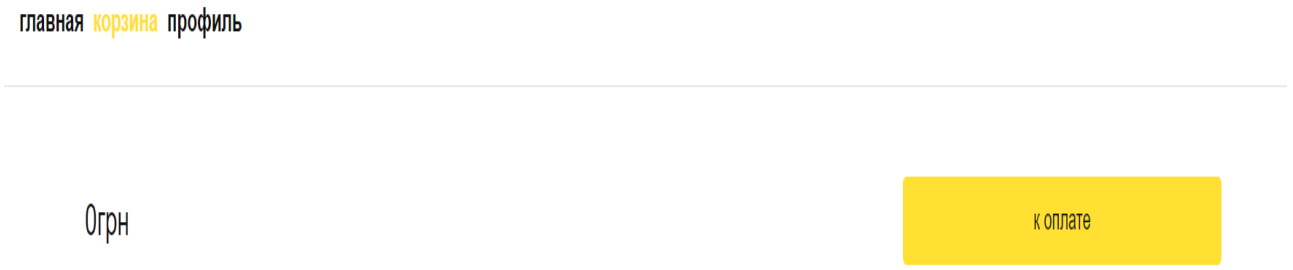
Коли ми реєструємося і входимо в обліковий запис, як споживач, перше, що ми можемо помітити - це вікно, яке пропонує нам усі товари на

вибір:



1.7. Початкове меню додатку

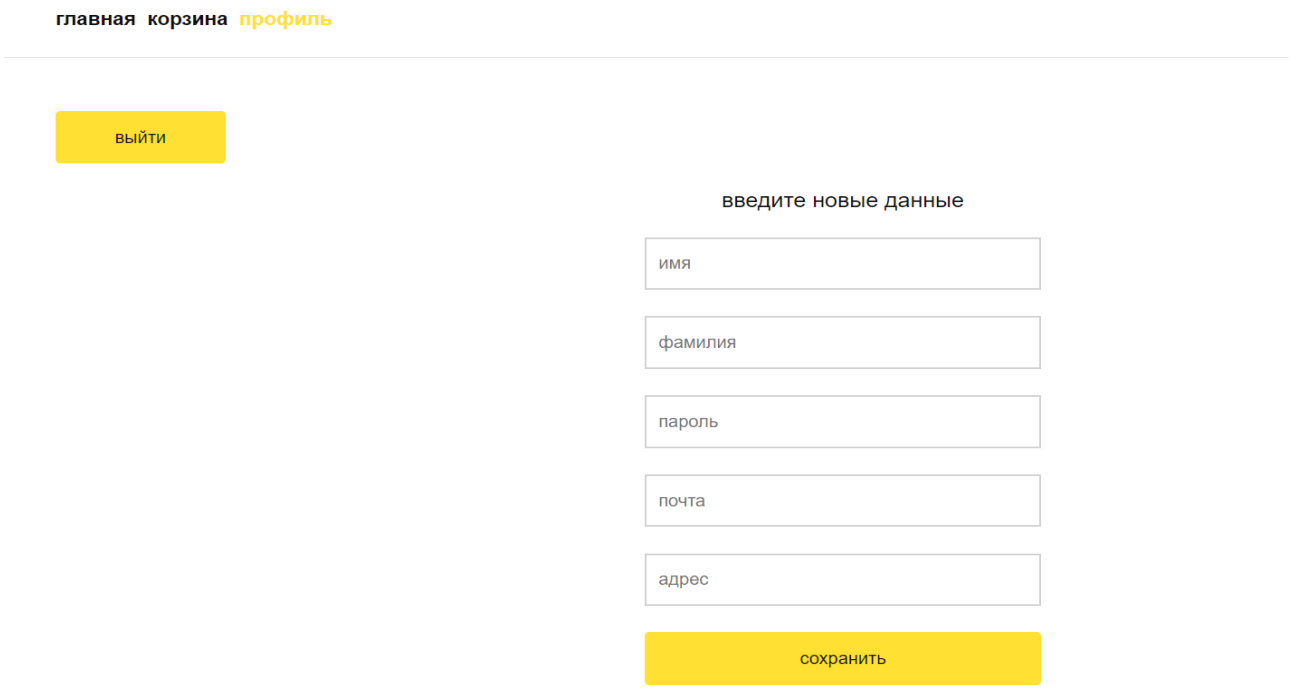
Також можна перейти до кошику, щоб переглянути товари, які були додані, збільшити або зменшити їх кількість, а також перейти до оплати замовлення, чекаючи доставки від кур'єра:



1.8. Меню корзинки

Є також розділ профілю, який дозволяє користувачеві змінювати необхідні йому дані. Зверху зліва також є кнопка виходу з облікового запису, після виходу з облікового запису користувач автоматично потрапляє в

початкове меню веб сайту:

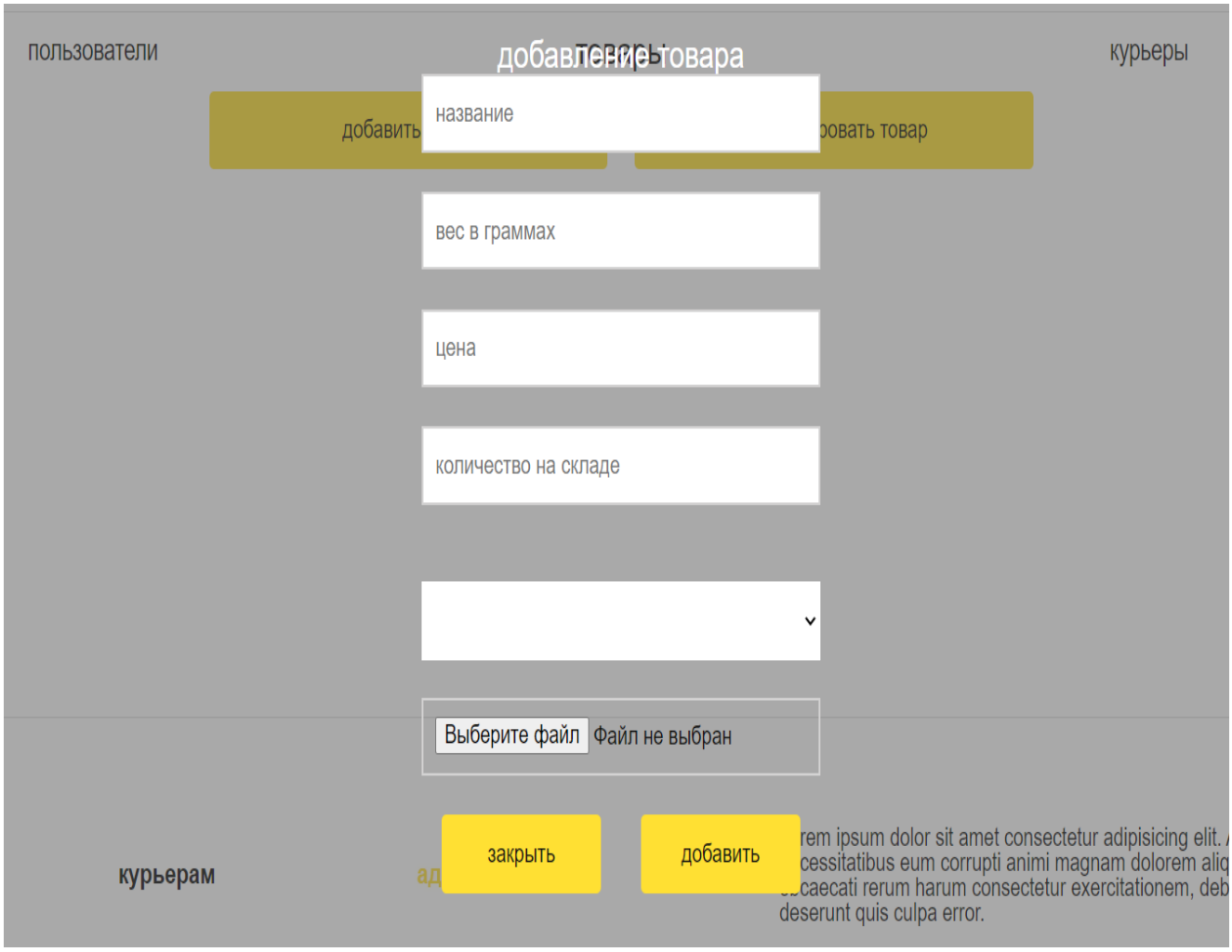


1.9. Меню профіля

Через аккаунт адміністратора можна впливати на всіх користувачів, які підключені до сервера, тобто можна додавати інших користувачів, а також видаляти.

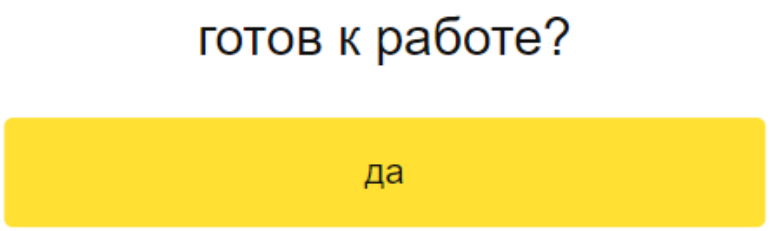
Також існує вкладка, яка дозволяє додавати товар користувачам, вказуючи назву, вагу в грамах, вартість продукту, а також кількість товарів, також можна вказувати фільтри і завантажувати картинку для відображення

товару:



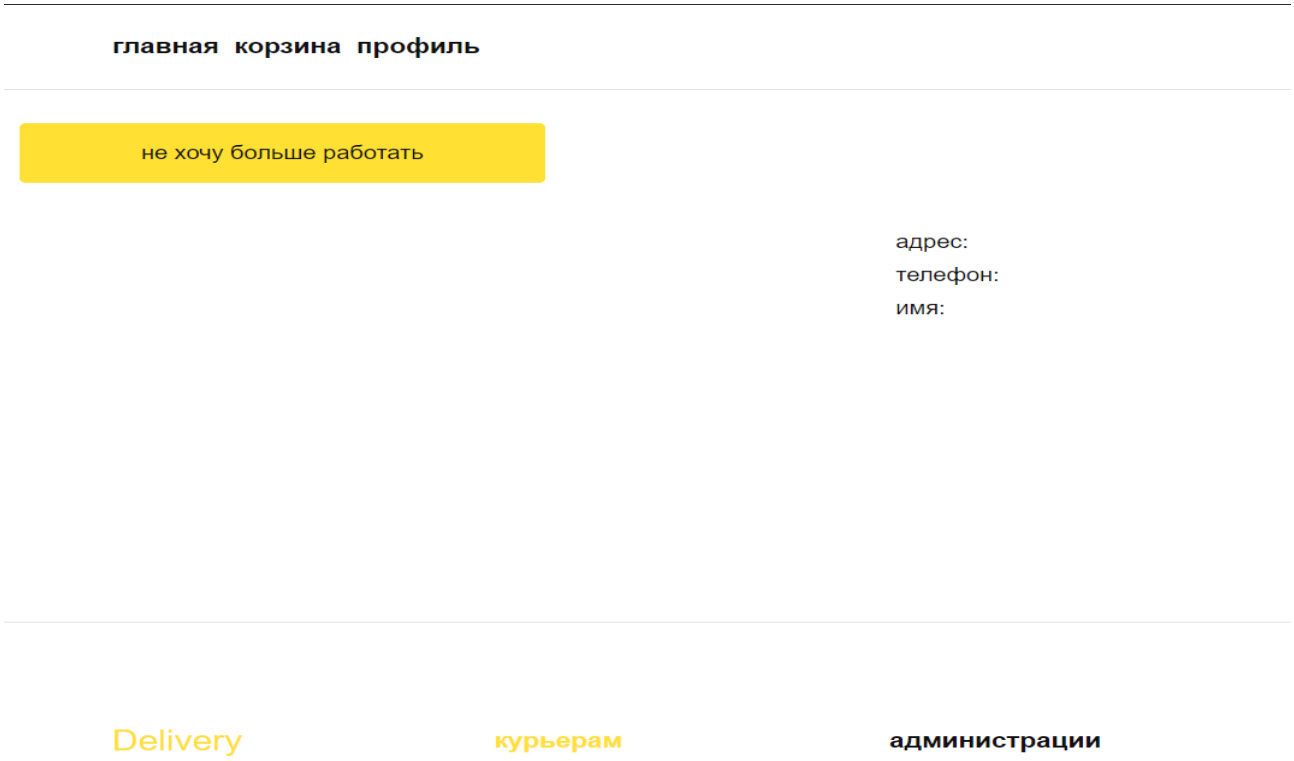
2.0. Меню додатку продуктів для споживачів

Доступна вкладка для кур'єрів, яка дозволяє почати роботу:



2.1. Меню кур’єра

У разі, якщо кур'єр вибирає "так", він автоматично переходить на наступну сторінку профілю в якій відображені адреса, телефон, а також ім'я. Також у кур'єрів доступна вкладка припинення роботи:



2.2. Меню кур’єра

РОЗДІЛ 3. ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Визначення трудомісткості розробки програмного забезпечення

При розробці програмного забезпечення (ПЗ) важливими етапами є визначення трудомісткості розробки і розрахунок витрат на створення програмного продукту.

Задані дані:

1) Передбачуване число операторів – 600;
2) Коефіцієнт складності програми – 1,5;
3) Коефіцієнт корекції програми в ході її розробки – 0,1;
4) Годинна заробітна плата програміста, грн/год – 250;
5) Коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,1;
6) Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,0;
7) Вартість машино-години ЕОМ, грн/год – 45.

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

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин.} \qquad (1.1)$$

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

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

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

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

$t_д$ - витрати праці на підготовку документації.

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

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \qquad (1.2)$$

Q - передбачуване число операторів;

C - коефіцієнт складності програми;

P - коефіцієнт кореляції програми в ході її розробки.

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

$$tu = Q * B(75..85) * k, \text{ людино-годин.} \qquad (1.3)$$

B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t\,a \cdot = Q/(20..25)*k, \text{ людино-годин.} \qquad (1.4)$$

Витрати на складання програми по готовій блок-схемі:

$$t\,n \cdot = Q/(20..25)*k, \text{ людино-годин.} \qquad (1.5)$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t\ отл \cdot = Q/(4..5)*k\ ,\ людино\text{-}годин. \qquad (1.6)$$

- за умови комплексного налагодження завдання:

$$Tk\ отл = 1{,}5 * t\ отл,\ людино\text{-}годин. \qquad (1.7)$$

Витрати праці на підготовку документації:

$$tд = tдр + tдо\ ,\ людино\text{-}годин, \qquad (1.8)$$

де tдр - трудомісткість підготовки матеріалів і рукопису.

$$t\ др \cdot = Q/(15..20)*k\ ,\ людино\text{-}годин \qquad (1.9)$$

tдо - трудомісткість редагування, печатки й оформлення документації.

$$Tk\ до = 0{,}75 * t\ др,\ людино\text{-}годин. \qquad (1.10)$$

3.2. Витрати на створення програмного забезпечення

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К\ по = З\ зп + З\ мв,\ грн. \qquad (1.11)$$

Заробітна плата виконавців визначається за формулою:

$$З\ зп = t \cdot С\ пр,\ грн. \qquad (1.12)$$

t - загальна трудомісткість, людино-годин;

С пр - середня годинна заробітна плата програміста, грн/година.

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{мв} = t_{отл} \cdot C_{мч}, грн. \quad (1.13)$$

Т отл - трудомісткість налагодження програми на ЕОМ, год.

С мч - вартість машино-години ЕОМ, грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ:

$$T = t / B_k * F_p, міс. \quad (1.14)$$

Bk - число виконавців;

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

ВИСНОВКИ

Метою дипломного проекту є розробка програмного забезпечення для доставки їжі споживачам, а також для користування додатком ресторанів чи власного підприємства. Програма являє собою систему, що реалізує запис інформації про замовлення користувачів, обробку та зберігання даних, відображення інформації у зрозумілій впорядкованій формі.

Мета даної дипломної роботи складалася з декількох частин, наприклад:

1) Написання плану дій дипломної роботи.

2) Здійснення задуманої концепції, яка відображається коректно, а також має подальше застосування в безпосередньо важливих сферах життя людини. Також додавання декількох функцій в процесі роботи, наприклад, таких як: зміна облікових даних, додавання або зміна будь-яких продуктів, включаючи їжу, але і також товари щоденного користування.

Тема дипломної роботи є актуальною на сьогоднішній день, а також, провівши певні аналітичні спостереження за розвитком сфери діяльності людини по доставці їжі, дана тема набирає свою актуальність і популярність. Існують емпіричні дані згідно з якими послуги з доставки будуть приносити більший прибуток в найближчому майбутньому. Доставка їжі хоч і не є новою і унікальною ідеєю, однак, допускає безліч нових поправок, а також коригувань і фільтрів, які дійсно можуть зробити продукт більш унікальним і універсальним, згідно смакам користувачів.

З практичної точки зору, доставка їжі буде займати в майбутньому лідируючі позиції, так велика частина людей замовляє вже готову їжу або продукти додому. Це пов'язано з ситуацією, яка виникла по всьому світу на

даний момент, а також з тим, що велика частина роботи зараз не мануальна, а технічна, тому багато хто замовляє їжу в офіси, на роботу, або рбаотают з дому.

Також, дана програма не є дорогим в плані його підтримки, проте, якщо пізніше враховувати кількість витрат на підтримку програми, вони можуть зрости. У додатку є кілька пунктів економічного прімінення, тобто воно буде застосовуватися від імені незалежної компанії, або ж використовуватися ресторанами і кафе. В кожному окремому випадку, даний вид додатка буде приносити величезний прибуток, вимагаючи мінімальні витрати. Прогноз для ситеми по доставці їжі позитивний, так як є діаграма згідно з якою, велика частина населення переходить на готове харчування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Джон Джаккет, Javascript и jQuery. Интерактивная веб-разработка, URL: https://monster-book.com/javascript-i-jquery-interaktivnaya-veb-razrabotka
2. Современный учебник JavaScript, URL: https://learn.javascript.ru/
3. Стефанов С, 124с. 75с. 89с, javascript. Шаблоны, URL: https://ru.pdfdrive.com/javascript-%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B-e186422699.html
4. Лоусон Б. Шарп Р, 68с, Изучаем HTML 5. Библиотека специалиста, URL: http://padabum.com/d.php?id=41242
5. П. Лабберс - HTML 5 для профессионалов URL: https://proklondike.net/books/html/labbers_html5.html
6. Дронов В. А. - HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов, URL: https://proklondike.net/books/webdesign/dronov_html5_css3_web20_2011.html
7. Бен Хеник - HTML и CSS Путь к совершенству, URL: https://proklondike.net/books/html/ben_henik_HTML_CSS_2011.html
8. Мэтт Стаффер, Laravel. Полное руководство, URL: https://monster-book.com/laravel-polnoe-rukovodstvo
9. Дмитрий Котеров, PHP 7 в подлиннике, URL: https://monster-book.com/php-7-v-podlinnike
10. Игорь Симдянов, Максим Кузнецов, Самоучитель PHP 7, URL: https://monster-book.com/samouchitel-php-7
11. Бретт Маклафлин, PHP и MySQL. Исчерпывающее руководство, URL: https://monster-book.com/php-i-mysql-ischerpyvayushchee-rukovodstvo

12. Яков Файн, Антон Моисеев, Angular и TypeScript. Сайтостроение для профессионалов, URL: https://monster-book.com/angular-i-typescript
13. Джон Дакетт, HTML и CSS. Разработка и дизайн веб-сайтов, URL: https://monster-book.com/html-i-css-razrabotka-i-dizayn-veb-saytov
14. Александр Чипижко, Прибыльная веб-студия. Пошаговое руководство, URL: https://monster-book.com/pribylnaya-veb-studiya-chipizhko
15. Кайл Симпсон, Типы и грамматические конструкции {Вы не знаете JS}, URL: https://monster-book.com/tipy-i-grammaticheskie-konstrukcii-js
16. Артур Чистяков, Как заработать на доставке еды, URL: https://monster-book.com/kak-zarabotat-na-dostavke-edy
17. Дейт К.Дж. Введение в системы баз данных 8-ое издание. — М.: Вильямс, 2005. — 1328 с., URL: https://www.twirpx.com/file/196117/
18. Джим Р. Уилсон, Эрик Редмонд, Семь баз данных за семь недель, URL: https://monster-book.com/7-baz-dannyh-za-7-nedel
19. Евгений Моргунов PostgreSQL. Основы языка SQL, URL: https://monster-book.com/postgresql-osnovy-yazyka-sql
20. Томас Коннолли, Каролин Бегг, Базы данных. Проектирование, реализация и сопровождение, URL: https://monster-book.com/bazy-dannyh-proektirovanie-realizaciya-i-soprovozhdenie
21. Джон Резиг, знания основ JavaScript. URL: http://null-book.com/?rout=article&article=941
22. Мартин Фаулер, Рефакторинг кода на JavaScript: улучшение проекта существующего кода, URL: https://revall.info/refaktoring-koda-na-javascript-uluchshenie-proekta-sushhestvuyushhego-koda.html

ДОДАТОК А

КОД ПРОГРАМИ

```
import { createRouter, createWebHistory } from 'vue-router'
import Home from '../views/Home.vue'

const routes = [
  {
    path: '/',
    name: 'Home',
    component: Home
  },
  {
    path: '/admin',
    name: 'Admin',
    // route level code-splitting
    // this generates a separate chunk (about.[hash].js) for this route
    // which is lazy-loaded when the route is visited.
    component: () => import(/* webpackChunkName: "about" */ '../views/admin/')
  },
  {
    path: '/postman',
    name: 'Postman',
    component: () => import('../views/postman/')
  },
  {
    path: '/auth',
    name: 'Auth',
    component: () => import('../views/auth/')
  },
  {
    path: '/profile',
    name: 'Profile',
    component: () => import('../views/user/')
  },
  {
    path: '/cart',
    name: 'Cart',
    component: () => import('../views/user/cart.vue')
  }
]

const router = createRouter({
  mode: "history",
  history: createWebHistory(process.env.BASE_URL),
  routes
```

```
})

export default router

import { createStore } from 'vuex'

export default createStore({
  state: {
    userData: {
      token: "",
    }

  },
  mutations: {
    setUserData(state, userData){
      state.userData = userData
    }
  },
  getters: {
    getUserData(state){
      return state.userData
    }
  },
})
```

1.1. Адмін

```
  template>
  <div class="admin">
    <nav class="nav__admin">
      <ul class="nav__list">
        <li class="nav__list-item users" @click="page = 1;changeSlide()">
          пользователи
        </li>
        <li class="nav__list-item products" @click="page = 2;changeSlide()">
          товары
        </li>
        <li class="nav__list-item postmans" @click="page = 3;changeSlide()">
          курьеры
        </li>
      </ul>
    </nav>
  <ul class="page__list">
    <li class="page__list-item">
      <div class="spacer"></div>
      <ul class="product__list">
        <li class="product__list-item" v-for="user in users" :key="user.id">
          <span>{{user.name}}</span>
          <span>{{user.lname}}</span>
          <span>{{user.phone}}</span>
          <span>{{user.email}}</span>
```

```
        <span v-if="user.rights == 1">администратор</span>
        <span v-else-if="user.rights == 2">курьер</span>
        <span v-else>пользователь</span>
        <span class="online" v-if="user.isactive == 1"></span>
        <span class="offline" v-else></span>

       </li>
      </ul>
     </li>
     <li class="page__list-item">
      <div class="popUpProduct">
       <div class="form">
        <span>добавление товара</span>
        <input type="text" placeholder="название" v-model="productToAdd.name">
        <input type="number" placeholder="вес в граммах" v-model="productToAdd.weight">
        <input type="number" placeholder="цена" v-model="productToAdd.cost">
        <input type="number" placeholder="количество на складе" v-model="productToAdd.qtyOnStorage">
        <select id="select">
         <option v-
for="category in categories" :key="category.id" :value="category.id">{{category.name}}</option>
        </select>
        <input id="fileinput" type="file" placeholder="картинка" accept="image/jpeg,image/png" required>
        <div>
         <button class="button" @click="popupClose()">закрыть</button>
         <button class="button" @click="addProduct()">добавить</button>
        </div>
       </div>
      </div>
      <div class="popUpProduct">
       <div class="form">
        <span>id товара для деактивации</span>
        <input type="number" placeholder="id товара" v-model="selectedProduct">
        <div>
         <button class="button" @click="popupClose()">закрыть</button>
         <button class="button" @click="removeProduct()">деактивировать</button>
        </div>
       </div>
      </div>
      <div class="page__header">
       <button class="button" @click="popupAddProduct()">добавить товар</button>
       <button class="button" @click="popupRemoveProduct()">деактивировать товар</button>
      </div>
      <ul class="product__list">
       <li class="product__list-item" v-for="product in products" :key="product.id">
        <img :src="'https://product-delivery.website'+product.imagepath" alt="">
        <p>{{product.name}}</p>
        <p>вес {{product.weight}}г.</p>
        <p>цена {{product.cost}}грн</p>
        <p>осталось {{product.qtyonstorage}} шт</p>
        <span class="offline" v-if="product.qtyonstorage == 0 || product.isactive == 0"></span>
        <span class="online"  v-else></span>
```

```
      </li>
     </ul>
    </li>
    <li class="page__list-item">
     <div class="spacer"></div>
     <ul class="product__list">
      <li class="product__list-item" v-for="postman in postmans" :key="postman.id">
       <span>{{postman.name}}</span>
       <span>{{postman.lname}}</span>
       <span>{{postman.phone}}</span>
       <span>{{postman.email}}</span>
       <span class="online" v-if="postman.isactive == 1"></span>
       <span class="offline" v-else></span>
      </li>
     </ul>
    </li>
   </ul>
   </div>
</template>

<script>
export default {
  name: 'Admin',
  data(){
    return{
      page: 1,
      postmans: {

      },
      users:{

      },
      products:{

      },
      selectedProduct: "",
      categories: [],
      productToAdd:{
        name: "",
        cost: "",
        weight: "",
        category: "",
        image: "",
        qtyOnStorage: ""
      }
    }
  },
  mounted(){
    if(this.$store.state.userData.token == ""){
      this.$router.push('auth')
```

```
  }else{
    document.querySelectorAll('.nav__list-item')[0].style.fontSize = '24px'
    document.querySelectorAll('.nav__list-item')[1].style.fontSize = '18px'
    document.querySelectorAll('.nav__list-item')[2].style.fontSize = '18px'
    document.querySelectorAll('.page__list-item')[0].style.display = 'flex'
    document.querySelectorAll('.page__list-item')[1].style.display = 'none'
    document.querySelectorAll('.page__list-item')[2].style.display = 'none'
    this.axios.post('https://product-delivery.website/api/api.php', {apiexecute: 11, token: localStorage.token}, {
      headers: {
          'Content-Type': 'multipart/form-data'
      }
    }).then((response) => {
      if(response.data[0].rights != 1){
        this.$router.push('auth')
      }
    })
    this.loadusers()
    this.axios.post('https://product-delivery.website/api/api.php', {apiexecute: 8}, {
      headers: {
          'Content-Type': 'multipart/form-data'
      }
    }).then((response) => {
      this.categories = response.data
      console.log(this.categories)
    })
  }
},
methods: {
  changeSlide(){
    if(this.page == 1){
      document.querySelectorAll('.page__list-item')[0].style.display = 'flex'
      document.querySelectorAll('.page__list-item')[1].style.display = 'none'
      document.querySelectorAll('.page__list-item')[2].style.display = 'none'
      document.querySelectorAll('.nav__list-item')[0].style.fontSize = '24px'
      document.querySelectorAll('.nav__list-item')[1].style.fontSize = '18px'
      document.querySelectorAll('.nav__list-item')[2].style.fontSize = '18px'
      this.loadusers()

    }else if(this.page == 2){
      document.querySelectorAll('.page__list-item')[0].style.display = 'none'
      document.querySelectorAll('.page__list-item')[1].style.display = 'flex'
      document.querySelectorAll('.page__list-item')[2].style.display = 'none'
      document.querySelectorAll('.nav__list-item')[0].style.fontSize = '18px'
      document.querySelectorAll('.nav__list-item')[1].style.fontSize = '24px'
      document.querySelectorAll('.nav__list-item')[2].style.fontSize = '18px'
      this.loadProducts()

    }else if(this.page == 3){
      document.querySelectorAll('.page__list-item')[0].style.display = 'none'
      document.querySelectorAll('.page__list-item')[1].style.display = 'none'
      document.querySelectorAll('.page__list-item')[2].style.display = 'flex'
```

```
    document.querySelectorAll('.nav__list-item')[0].style.fontSize = '18px'
    document.querySelectorAll('.nav__list-item')[1].style.fontSize = '18px'
    document.querySelectorAll('.nav__list-item')[2].style.fontSize = '24px'
    this.loadPostmans()
  }
},
loadPostmans(){
  this.axios.post('https://product-delivery.website/api/api.php', {apiexecute: 12, token: localStorage.token}, {
    headers: {
        'Content-Type': 'multipart/form-data'
    }
  }).then((response) => {
    this.postmans = response.data
  })
},
loadusers(){
  this.axios.post('https://product-delivery.website/api/api.php', {apiexecute: 13, token: localStorage.token}, {
    headers: {
        'Content-Type': 'multipart/form-data'
    }
  }).then((response) => {
    this.users = response.data
  })
},
loadProducts(){
  this.axios.post('https://product-delivery.website/api/api.php', {apiexecute: 14, token: localStorage.token}, {
    headers: {
        'Content-Type': 'multipart/form-data'
    }
  }).then((response) => {
    this.products = response.data
  })
},
popupClose(){
  document.querySelectorAll('.popUpProduct')[0].style.display = "none"
  document.querySelectorAll('.popUpProduct')[1].style.display = "none"
},
popupAddProduct(){
  document.querySelectorAll('.popUpProduct')[0].style.display = "grid"
},
popupRemoveProduct(){
  document.querySelectorAll('.popUpProduct')[1].style.display = "grid"
},
addProduct(){
  this.productToAdd.category = document.getElementById('select').value
  let formdata = new FormData()
  formdata.append('apiexecute', 15)
  formdata.append('name',this.productToAdd.name)
  formdata.append('cost',this.productToAdd.cost)
  formdata.append('weight',this.productToAdd.weight)
  formdata.append('category',this.productToAdd.category)
```

```
      formdata.append('qtyonstorage',this.productToAdd.qtyOnStorage)
      formdata.append('image', document.getElementById('fileinput').files[0], document.getElementById('fileinput')
.files[0].name)
      this.axios.post('https://product-delivery.website/api/api.php', formdata, {
        headers: {
            'Content-Type': 'multipart/form-data'
        }
      }).then((response) => {
        this.popupClose()
        this.loadProducts()
      })
    },
    removeProduct(){
      this.axios.post('https://product-delivery.website/api/api.php', {apiexecute: 16, product: this.selectedProduct}, {
        headers: {
            'Content-Type': 'multipart/form-data'
        }
      }).then((response) => {
        this.popupClose()
        this.loadProducts()
      })
    }
  }

}
</script>
<style lang="scss" scoped>
.nav__admin{
  .nav__list{
      display: flex;
      align-items: center;
      justify-content: space-around;
    .nav__list-item{
      text-align: center;
      width: 150px;
      height: 50px;
      display: flex;
      justify-content: center;
      align-items: center;
      font-size: 18px;
      font-weight: 400;
      cursor: pointer;
      user-select: none;
    }
  }
}
.product__list{
  display: flex;
  flex-direction: column;
  align-items: center;
  width: 100%;
```

```
  margin-top: 50px;
  .product__list-item{
    width: 100%;
    height: 50px;
    margin-bottom: 25px;
    font-size: 16px;
    display: grid;
    place-items: center;
    grid-template-columns: 1fr 1fr 1fr 1fr 1fr 0.1fr;
    padding: 0 50px;
    color: #101010;
    .online{
      background: rgb(108, 199, 24);
      border-radius: 50%;
      height: 10px;
      width: 10px;
    }
    .offline{
      background: rgb(201, 69, 16);
      border-radius: 50%;
      height: 10px;
      width: 10px;
    }
    img{
      height: 50px;
      width: 50px;
    }
  }
}
.page__list-item{
  display: none;
  flex-direction: column;
  justify-content: space-around;
  align-items: center;
  .spacer{
    height: 50px;
  }
  .page__header{
    display: flex;
    justify-content: space-between;
  }
}
.button{
    font-size: 16px;
    background: #ffe033;
    color: #101010;
    height: 50px;
    width: 350px;
    border-radius: 4px;
    transition: 0.2s;
    margin: 0 12px;
```

```
    &:hover{
      color: #fff;
      background: #101010;
    }
}
.popUpProduct{
  width: 100vw;
  height: 100vh;
  position: fixed;
  top: 0;
  left: 0;
  display: none;
  place-items: center;
  z-index: 9999;
  background: rgba(83, 83, 83, 0.5);
  .form{
    width: 350px;
    display: flex;
    flex-direction: column;
    span{
      text-align: center;
      font-size: 24px;
      color: #fff;
    }
    select{
      margin: 24px 0;
      height: 50px;
    }
    div{
      display: flex;
    }
    .button{
      margin: 0 5%;
      width: 40%;
    }
  }
}
input{
  height: 50px;
  padding: 10px;
  border: 2px solid #d3d2d2;
  margin-bottom: 25px;
  color: #101010;
  font-size: 16px;
  font-family: arial;
}
</style>
```

1.2. Регістрація на сайті

```
<template>
```

```
  <div class="auth">
    <div class="auth-wrapp">
      <form class="auth__form-signin" @submit.prevent="this.setupSignInData()">
        <input class="input" type="tel" placeholder="телефон" v-model="this.phone">
        <input class="input" type="password" placeholder="пароль" v-model="this.password">
        <button class="button" type="submit">войти</button>
        <div class="auth-button" @click="setupForm()">зарегистрироваться</div>
      </form>
      <form class="auth__form-signup" @submit.prevent="this.setupSignUpData()">
        <input class="input" type="text" placeholder="имя" v-model="this.name">
        <input class="input" type="text" placeholder="фамилия" v-model="this.lastname">
        <input class="input" type="password" placeholder="пароль" v-model="this.password">
        <input class="input" type="password" placeholder="повторите пароль" v-model="this.repeatpass">
        <input class="input" type="text" placeholder="почта" v-model="this.email">
        <input class="input" type="tel" placeholder="телефон" v-model="this.phone">
        <button class="button" type="submit">зарегистрироваться</button>
        <div class="auth-button" @click="setupForm()">войти</div>
      </form>
    </div>
  </div>
</template>

<script>
export default {
  name: 'auth',
  data(){
    return{
      islogin: false,
      password: "",
      repeatpass: "",
      email: "",
      phone: "",
      name: "",
      lastname: "",
    }
  },
  mounted(){
    if(localStorage.token){
      this.$router.push('profile')
    }
  },
  methods: {
    setupForm: function () {
      if(this.islogin){
        document.querySelector('.auth__form-signin').style.left = "50%"
        document.querySelector('.auth__form-signup').style.left = "150%"
        document.querySelector('.auth__form-signup').style.opacity = "0"
        document.querySelector('.auth__form-signin').style.opacity = "1"
        this.login = ""
        this.password = ""
        this.repeatpass = ""
```

```
    this.email = ""
    this.phone = ""
    this.name = ""
    this.lastname = ""
    this.islogin = !this.islogin
   }else{
    document.querySelector('.auth__form-signin').style.left = "-150%"
    document.querySelector('.auth__form-signup').style.left = "50%"
    document.querySelector('.auth__form-signup').style.opacity = "1"
    document.querySelector('.auth__form-signin').style.opacity = "0"
    this.login = ""
    this.password = ""
    this.repeatpass = ""
    this.email = ""
    this.phone = ""
    this.name = ""
    this.lastname = ""
    this.islogin = !this.islogin
   }
  },
  setupSignInData: function(){
   let formdata = new FormData();
   formdata.append('apiexecute', 1);
   formdata.append('password', this.password);
   formdata.append('phone', this.phone);
   console.log(this.password);
   this.axios.post('https://product-delivery.website/api/api.php', formdata, {
    headers: {
      'Content-Type': 'multipart/form-data'
    }
   }).then((response) => {
    if(!response.data.errors){
     this.$store.state.userData.token = response.data.token
     localStorage.token = response.data.token
     this.$router.push('cart')
    }

   })
  },
  setupSignUpData: function(){
   let formdata = new FormData();
   formdata.append('apiexecute', 2);
   formdata.append('name', this.name);
   formdata.append('lname', this.lastname);
   formdata.append('email', this.email);
   formdata.append('password', this.password);
   formdata.append('phone', this.phone);
   console.log(this.password);
   this.axios.post('https://product-delivery.website/api/api.php', formdata, {
    headers: {
      'Content-Type': 'multipart/form-data'
```

```
        }
      }).then((response) => {
        console.log(response.data)
        if(!response.data.errors){
          this.$store.state.userData.token = response.data.token
          localStorage.token = response.data.token
          this.$router.push('cart')
        }
      })
    }
  }
}
</script>

<!-- Add "scoped" attribute to limit CSS to this component only -->
<style lang="scss">
#app{
  padding: 0 !important;
}
.auth{
  display: grid;
  place-items: center;
  height: 100%;
  width: 100%;
  .auth-wrapp{
    width: 1000px;
    height: 640px;
    overflow: hidden;
    position: relative;
    .input{
      height: 50px;
      padding: 10px;
      border: 2px solid #d3d2d2;
      margin-bottom: 25px;
      color: #101010;
      font-size: 16px;
      font-family: arial;
    }
    .button{
      height: 50px;
      min-width: 150px;
      color: #fff;
      background: #101010;
      font-family: arial;
      font-size: 16px;
    }
    .auth-button{
      margin: 25px 0;
      cursor: pointer;
    }
    .auth__form-signin{
```

```
    display: flex;
    flex-direction: column;
    width: 350px;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    transition: 0.3s;
   }
   .auth__form-signup{
    display: flex;
    flex-direction: column;
    width: 350px;
    position: absolute;
    top: 50%;
    left: 150%;
    transform: translate(-50%, -50%);
    transition: 0.3s;
   }
  }
}
</style>
```

1.3. Кур’єр

```
<template>
  <div class="postman">
    <div class="ready" v-if="ready == false">
      <span>готов к работе?</span>
      <button class="button" @click="setReady()">да</button>
    </div>
    <div class="order" v-else>
      <p>адрес: {{adress}}</p>
      <p>телефон: {{phone}}</p>
      <p>имя: {{name}}</p>
      <ul class="order__list">
        <li class="order__list-item" v-for="item in order" :key="item.id">
          <span>{{item.id}}.</span>
          <span>{{item.productname}}</span>
          <span>{{item.qty}}шт.</span>
        </li>
        <button class="button" @click="closeOrder()" v-
if="ready != false && order.length != 0">завершить заказ</button>
      </ul>
    </div>
    <button class="button unready" @click="setUnready()" v-
if="ready != false && order.length == 0">не хочу больше работать</button>
  </div>
</template>

<script>
```

```
export default {
  name: 'Postman',
  data (){
    return {
      ready: false,
      order: [],
      name: "",
      phone: "",
      adress: ""
    }
  },
  mounted(){
    if(this.$store.state.userData.token == ""){
      this.$router.push('auth')
    }else{
      this.axios.post('https://product-delivery.website/api/api.php', {apiexecute: 11, token: localStorage.token}, {
        headers: {
            'Content-Type': 'multipart/form-data'
        }
      }).then((response) => {
        if(response.data[0].rights != 2){
          this.$router.push('auth')
        }else{
          if(localStorage.ready){
            this.ready = localStorage.ready
            this.setReady()
          }else{
            this.ready = false
          }
        }
      })
    }
  },
  methods: {
    setReady(){
      this.ready = true
      localStorage.ready = true
      this.axios.post('https://product-delivery.website/api/api.php',  {apiexecute: 17, token: localStorage.token}, {
        headers: {
            'Content-Type': 'multipart/form-data'
        }
      }).then((response) => {
        if(response.data.length != 0){
          this.order = response.data
          console.log(response.data);
          this.phone = this.order[0].phone
          this.adress = this.order[0].adress
          this.name = this.order[0].name
        }
      })
    },
```

```
    setUnready(){
      this.ready = false
      localStorage.ready = false
    },
    closeOrder(){
      this.axios.post('https://product-delivery.website/api/api.php',  {apiexecute: 19, token: localStorage.token}, {
        headers: {
            'Content-Type': 'multipart/form-data'
        }
      }).then((response) => {
        this.order = []
        this.setReady()
      })
    }
  }
}
</script>
<style lang="scss" scoped>
.unready{
  position: absolute;
  left: 10px;
  top: 100px;
}
.button{
  display: grid;
  place-items: center;
  font-size: 16px;
  background: #ffe033;
  color: #101010;
  height: 50px;
  width: 350px;
  border-radius: 4px;
  transition: 0.2s;
  cursor: pointer;
  &:hover{
    color: #fff;
    background: #101010;
  }
}
.ready{
  font-size: 24px;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  display: flex;
  flex-direction: column;
  text-align: center;
  .button{
    margin-top: 25px;
  }
```

```
}
.order{
  font-size: 16px;
  margin-top: 120px;
  margin-left: 50%;
  transform: translateX(-50%);
  display: flex;
  flex-direction: column;
  width: 350px;
  p{
    margin-bottom: 12px;
  }
}
.order__list{
  display: flex;
  flex-direction: column;
  .order__list-item{
    border-radius: 4px;
    display: flex;
    align-items: center;
    margin-bottom: 12px;
    background: #e2e2e2;
    height: 35px;
    padding: 0 12px;
    span:nth-child(1){
      margin-right: 10px;
    }
    span:nth-child(2){
      margin-right: auto;
    }
  }
}
</style>
```

1.4. Оплата

```
<template>
  <div class="cart">
    <div class="cart-item" v-if="!cart">
      <span>корзина пуста</span>
    <router-link class="button" to="/">начать покупки!</router-link>
    </div>
    <div v-else>
      <ul class="cart__list">
        <li class="cart__list-item" v-for="cartproduct in cartproducts" :key="cartproduct.id">
          <img class="list__item-image" :src="'https://product-delivery.website'+cartproduct.imagepath" alt="">
          <p class="list__item-name">{{cartproduct.name}}</p>
          <p class="list__item-cost">{{cartproduct.cost}}грн</p>
          <p class="list__item-qty">{{cartproduct.qty}}</p>
          <div class="list__block-qty">
            <button class="item__qty-prev" @click="removeQtyProduct(cartproduct.id)">-</button>
```

```
          <button class="item__qty-next" @click="addQtyProduct(cartproduct.id)">+</button>
         </div>
        </li>
       </ul>
       <div class="submit">
        <div class="submit__cost"></div>
        <button class="submit__button" @click="makePayment()">к оплате</button>
       </div>
     </div>

   </div>
</template>

<script>
export default {
  name: 'Cart',
  data (){
    return {
      cart: true,
      postbody: {
        apiexecute: "3",
        token: localStorage.token
      },
      cartproducts: []
    }
  },
  methods: {
    setlastCost: function(){
      let cost = 0
      for (let i = 0; i < this.cartproducts.length; i++) {
        cost += parseInt(this.cartproducts[i].cost*this.cartproducts[i].qty)
      }
      document.querySelector('.submit__cost').innerText = cost+"грн"
    },
    addQtyProduct: function(id){
      this.axios.post('https://product-
delivery.website/api/api.php',  {apiexecute: 6, token: localStorage.token, idproduct: id}, {
        headers: {
          'Content-Type': 'multipart/form-data'
        }
      }).then((response) => {
        this.getCartData()
      })

    },
    removeQtyProduct: function(id){
      this.axios.post('https://product-
delivery.website/api/api.php', {apiexecute: 7, token: localStorage.token, idproduct: id}, {
      headers: {
        'Content-Type': 'multipart/form-data'
      }
```

```
    }).then((response) => {
      this.getCartData()
    })

    },
    getCartData: function(){
      this.axios.post('https://product-delivery.website/api/api.php', this.postbody, {
        headers: {
            'Content-Type': 'multipart/form-data'
        }
      }).then((response) => {
        if(response.data.cart == 0){
          this.cart = false
        }else{
          this.cartproducts = response.data
          this.cart = true
          this.setlastCost()
        }
      })
    },
    makePayment: function(){
      this.axios.post('https://product-delivery.website/api/api.php', {token: localStorage.token, apiexecute: 18}, {
        headers: {
            'Content-Type': 'multipart/form-data'
        }
      }).then((response) => {
        this.cartproducts = []
        this.getCartData()
      })
    }
  },
  mounted(){
    if(this.$store.state.userData.token == ""){
      this.$router.push('auth')
    }else{
      this.getCartData()
    }
  },

}
</script>
<style lang="scss" scoped>
.cart{
  display: flex;
  flex-direction: column;
  height: 100%;
  width: 100%;
  flex: 1 0 auto;
  .cart-item{
    justify-content: center;
    align-items: center;
```

```
    height: 100%;
    width: 100%;
    display: flex;
    flex-direction: column;
    span{
      margin-top: 25%;
      font-size: 50px;
      color: #e2e2e2;
      transform: translateY(-50%);
    }
  }
}
.cart__list{
  flex: 1 0 auto;
  .cart__list-item{
    display: flex;
    justify-content: space-between;
    align-items: center;
    height: 50px;
    margin-top: 12px;
    padding: 0 72px;
    .list__item-image{
      height: 40px;
      width: 40px;
    }
    .list__item-qty{
      margin-left: 35%;
    }
    .list__block-qty{
      display: flex;
      .item__qty-prev{
        font-size: 30px;
        color: #101010;
        background: #ffe033;
        display: grid;
        place-items: center;
        border-radius: 4px;
        height: 35px;
        width: 35px;
        margin-right: 12px;
        line-height: 20px;
      }
      .item__qty-next{
        font-size: 30px;
        color: #101010;
        background: #ffe033;
        display: grid;
        place-items: center;
        border-radius: 4px;
        height: 35px;
        width: 35px;
```

```
      }
    }
  }
}
.submit{
  padding: 0 72px;
  flex: 0 0 auto;
  width: 100%;
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-top: 50px;
  .submit__cost{
    color: #101010;
    font-size: 24px;
  }
  .submit__button{
    font-size: 16px;
    background: #ffe033;
    color: #101010;
    height: 50px;
    width: 350px;
    border-radius: 4px;
    transition: 0.2s;
    &:hover{
      color: #fff;
      background: #101010;
    }
  }
}
.button{
  font-size: 16px;
  background: #ffe033;
  color: #101010;
  height: 50px;
  width: 350px;
  border-radius: 4px;
  transition: 0.2s;
  display: grid;
  place-items: center;
  &:hover{
    color: #fff;
    background: #101010;
  }
}
</style>
```

1.5. Користувач

```
<template>
  <div class="user">
```

```
    <div class="user__header">
      <a class="button" @click="exit()">выйти</a>
    </div>
    <div class="user__form">
      <div class="wrapp">
        <span>введите новые данные</span>
        <input type="text" placeholder="имя" v-model="userData.name">
        <input type="text" placeholder="фамилия" v-model="userData.lname">
        <input type="text" placeholder="пароль" v-model="userData.password">
        <input type="text" placeholder="почта" v-model="userData.email">
        <input type="text" placeholder="адрес" v-model="userData.adress">
        <button class="button" @click="setUserData()">сохранить</button>
      </div>
    </div>
  </div>
</template>

<script>
export default {
  name: 'User',
  data (){
    return {
      userData: {
        name: "",
        email: "",
        password: "",
        lname: "",
        adress: "",
      }
    }
  },
  mounted(){
    if(this.$store.state.userData.token == ""){
      this.$router.push('auth')
    }else{
      this.getUserData()
    }
  },
  methods: {
    exit: function(){
      localStorage.clear('token')
      localStorage.clear('ready')
      this.$store.state.userData.token = ""
      this.$router.push('auth')
    },
    getUserData: function(){
      this.axios.post('https://product-delivery.website/api/api.php',  {apiexecute: 9, token: localStorage.token}, {
        headers: {
            'Content-Type': 'multipart/form-data'
        }
      }).then((response) => {
```

```
        this.userData.name = response.data[0].name
        this.userData.lname = response.data[0].lname
        this.userData.email = response.data[0].email
        this.userData.adress = response.data[0].adress
        this.userData.password = ""
      })
    },
    setUserData: function(){
      this.axios.post('https://product-
delivery.website/api/api.php',  {apiexecute: 10, token: localStorage.token, name: this.userData.name, lname:this.u
serData.lname, email: this.userData.email, adress: this.userData.adress, password: this.userData.password}, {
        headers: {
            'Content-Type': 'multipart/form-data'
        }
      }).then((response) => {
        console.log(response.data);
        this.getUserData()
      })
    },
  }
}
</script>
<style lang="scss" scoped>
.user__header{
  margin-top: 50px;
}
input{
  height: 50px;
  padding: 10px;
  border: 2px solid #d3d2d2;
  margin-bottom: 25px;
  color: #101010;
  font-size: 16px;
  font-family: arial;
}
.user__header{
  .button{
    width: 150px;
  }
}
.user__form{
  display: grid;
  place-items: center;
  margin-top: 25px;
}
.wrapp{
  display: flex;
  flex-direction: column;
  width: 350px;
  span{
    font-size: 20px;
```

```
    margin-bottom: 25px;
    text-align: center;
  }
}
.button{
  display: grid;
  place-items: center;
  font-size: 16px;
  background: #ffe033;
  color: #101010;
  height: 50px;
  width: 350px;
  border-radius: 4px;
  transition: 0.2s;
  cursor: pointer;
  &:hover{
    color: #fff;
    background: #101010;
  }
}
</style>
```

1.6. Початкове меню

```
<template>
  <div class="home">
    <div class="categories">
      <ul class="categories__list">
        <li class="categories__list-item" @click="selectedCategory = 0;setUpProducts()">все</li>
        <li class="categories__list-item" v-
for="category in categories" :key="category.id" @click="selectedCategory = category.id;setUpProducts()">{{cat
egory.name}}</li>
      </ul>
    </div>
    <div class="store">
      <ul class="store__list">
        <li class="store__list-item" v-for="product in products" :key="product.id">
          <div class="store__item-header">
            <div class="item__header-info" :title="'id товара: '+product.id"></div>
          </div>
          <div class="store__item-body">
            <img class="item__body-image" :src="'https://product-delivery.website'+product.imagepath">
            <div class="item__body-description">
              <p class="item__description-title">{{product.name}}</p>
              <p class="item__description-weight">вес: <span>{{product.weight}}г.</span></p>
            </div>
          </div>
          <div class="store__item-footer">
            <p class="item__footer-cost">{{product.cost}}грн</p>
            <div class="item__footer-wrapp">
```

```
            <button class="item__footer-button" v-if="product.qty" @click="removeProduct(product.id)">-
</button>
            <span>{{product.qty}}</span>
            <button class="item__footer-button" @click="addProduct(product.id)">+</button>
           </div>
          </div>
        </li>
      </ul>
    </div>
  </div>
</template>

<script>
export default {
  name: 'Home',
  data (){
    return {
      postbody: {
        apiexecute: 5,
        category: 0,
        token: "",
      },
      addToCart: {
        idproduct: "",
        apiexecute: 6,
        token: "",
        qty: 1
      },
      products: [],
      categories: [],
      selectedCategory: 0,
    }
  },
  mounted(){
    this.axios.post('https://product-delivery.website/api/api.php', {apiexecute: 8}, {
      headers: {
          'Content-Type': 'multipart/form-data'
      }
    }).then((response) => {
      this.categories = response.data
    })
    this.setUpProducts()

  },
  methods:{
    addProduct: function(id){
      if(localStorage.token){
        this.axios.post('https://product-
delivery.website/api/api.php',  {apiexecute: 6, token: localStorage.token, idproduct: id}, {
          headers: {
              'Content-Type': 'multipart/form-data'
```

```
        }
       }).then((response) => {
        this.setUpProducts()
       })
      }else{
       this.$router.push('auth')
      }
    },
    removeProduct: function(id){
      this.axios.post('https://product-
delivery.website/api/api.php', {apiexecute: 7, token: localStorage.token, idproduct: id}, {
      headers: {
          'Content-Type': 'multipart/form-data'
      }
      }).then((response) => {
       this.setUpProducts()
      })
    },
    setUpProducts: function(){
      this.postbody.category = this.selectedCategory
      this.postbody.token = localStorage.token
      if(localStorage.token == undefined){
       this.postbody.token = ""
      }
      console.log(this.postbody);
      this.axios.post('https://product-delivery.website/api/api.php', this.postbody, {
       headers: {
          'Content-Type': 'multipart/form-data'
       }
      }).then((response) => {
       console.log(response.data);
       this.products = response.data
      })
    },
  }
}
</script>
<style lang="scss" scoped>

.categories{
  margin-top: 10px;
  .categories__list{
    display: flex;
    flex-wrap: wrap;
    margin: 0 auto;
    // justify-content: space-between;
    .categories__list-item{
      height: 175px;
      width: 175px;
      display: grid;
      place-items: center;
```

```
      border: 1px solid #e2e2e2;
      margin: 12px;
      transition: 0.1s;
      cursor: pointer;
      font-size: 18px;
      &:hover{
        background: #ffe033;
        border: 1px solid #ffe033;
      }
    }
  }
}
.store{
  width: 100%;
  height: auto;
  .store__list{
    display: flex;
    // justify-content: space-between;
    flex-wrap: wrap;
    margin: 0 auto;
    .store__list-item{
      position: relative;
      width: 250px;
      min-height: 300px;
      margin: 12px;
      padding: 8px;
      transition: 0.3s;
      box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.13);
      border-radius: 4px;
      display: flex;
      justify-content: space-between;
      flex-direction: column;
      .store__item-header{
        position: relative;
          height: 20px;
        .item__header-info{
          position: absolute;
          top: 0;
          right: 0;
          height: 20px;
          width: 20px;
          border-radius: 50%;
          border: 1px solid #e2e2e2;
          cursor: pointer;
        }
      }
      .store__item-body{
        .item__body-image{
          height: 150px;
          width: 100%;
          margin: 10px 0;
```

```
  }
  .item__body-description{
    display: flex;
    flex-direction: column;
    width: 100%;
    .item__description-title{
      font-size: 18px;
      font-weight: 700;

    }
    .item__description-weight{
      margin-top: 5px;
      font-size: 14px;
      color: #e2e2e2;

    }
  }
}
.store__item-footer{
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-top: 30px;
  .item__footer-wrapp{
    display: flex;
    justify-content: space-between;
    align-items: center;
    span{
      margin: 0 6px;
    }
  }
  .item__footer-cost{
    font-size: 24px;
    font-weight: 900;
  }
  .item__footer-button{
    height: 35px;
    width: 35px;
    background: #ffe033;
    box-shadow: 0 0 10px rgba(255, 224, 51, 0.13);
    display: grid;
    place-items: center;
    font-size: 24px;
    color: #101010;
    border-radius: 4px;
    transition: 0.1s;
    &:hover{
      background: #101010;
      color: #fff;
    }
  }
```

```
    }
   }
   .store__list-item:hover{
    transform: scale(1.015);
    box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.05);
   }
  }

}
</style>
```

1.7. Додаток

```
<template>
  <div id="nav">
    <router-link to="/">главная</router-link>
    <router-link to="/cart">корзина</router-link>
    <router-link to="/profile">профиль</router-link>
  </div>
  <div class="content">
    <router-view/>
  </div>

  <div class="footer">
    <p class="logo">delivery</p>
    <router-link to="/postman">курьерам</router-link>
    <router-link to="/admin">администрации</router-link>
    <p class="footer__description">Lorem ipsum dolor sit amet consectetur adipisicing elit. Accusamus laborum n
ecessitatibus eum corrupti animi magnam dolorem aliquid odio quam obcaecati rerum harum consectetur exercitat
ionem, debitis, quisquam deserunt quis culpa error.</p>
  </div>
</template>

<script>
export default {
  data (){
    return {
      token: false,
      postbody: {
        apiexecute: "4"
      }
    }
  },
  mounted(){
    if(localStorage.token){
      this.$store.commit('setUserData', [localStorage.token, 'name', 'lname', 'path'])
      this.token = true
    }
  },
}
</script>
```

```
<style lang="scss">
*{padding: 0;margin: 0;border: 0; scroll-behavior: smooth;}
*,*:before,*:after{-moz-box-sizing: border-box;-webkit-box-sizing: border-box;box-sizing: border-box;}
:focus,:active{outline: none;}
a:focus,a:active{outline: none;}
nav,footer,header,aside{display: block;}
html,body{height:100%;width:100%;font-size:100%;line-height:1;font-size:14px;-ms-text-size-adjust:100%;-
moz-text-size-adjust:100%;-webkit-text-size-adjust:100%;}
input,button,textarea{font-family:inherit;}
input::-ms-clear{display: none;}
button{cursor: pointer;user-select: none;}
button::-moz-focus-inner{padding:0;border:0;}
a,a:visited{text-decoration: none;}
a:hover{text-decoration: none;}
ul li{list-style: none;}
img{vertical-align: top;}
h1,h2,h3,h4,h5,h6{font-size:inherit;font-weight: inherit;}
@media (min-width: 500px) {
   ::-webkit-scrollbar{
      width: 30px;
   }
   ::-webkit-scrollbar-track{
      border-radius: 100px;
      background-clip: border-box;
   }
   ::-webkit-scrollbar-thumb{
      background: rgba(0, 0, 0, 0.5);
      background-clip: padding-box;
      border: 13px solid rgba(0, 0, 0, 0.0);
      border-radius: 100px;
   }
   ::-webkit-scrollbar-track-piece{
      background: rgba(0, 0, 0, 0.0);
   }
   ::-webkit-scrollbar-button{
      background: rgba(0, 0, 0, 0.0);
   }
   ::-webkit-scrollbar-corner {
      background: rgba(0, 0, 0, 0.0);
   }
   .content{
     padding: 72px 72px 0 72px !important;
   }
}
@media (max-width: 600px) {
  .footer__description{
    font-size: 14px !important;
  }
  .footer{
    height: 300px !important;
```

```
  }
}
body{
  min-height: 100vh;
  width: 100vw;
  overflow-y: overlay;
  overflow-x: hidden;
}
#app {
  font-family: Helvetica;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  color: #101010;
  min-height: 100vh;
  width: 100vw;
  display: flex;
  flex-direction: column;
}
.logo{
  font-size: 24px;
  color: #ffe033;
  text-transform: capitalize;
}
#nav {
  width: 100vw;
  position: fixed;
  top: 0;
  left: 0;
  height: 72px;
  padding: 0 72px;
  display: flex;
  align-items: center;
  background: #ffffff;
  border-bottom: 1px solid #e2e2e2;
  z-index: 999;
  a {
    font-size: 18px;
    color: #101010;
    margin-right: 10px;
    font-weight: 600;
    transition: 0.2s;
    &::after{
      height: 1px;
      background: #e2e2e2;
      width: 100%;
    }
    &:hover{
      color: #e2e2e2;
    }
    &.router-link-exact-active {
```

```
      color: #ffe033;
      text-shadow: 0 0 5px rgba(255, 224, 51, 0.13);
      // box-shadow: 0 0 10px rgba(255, 224, 51, 0.13);
    }
  }
}
.content{
  width: 100vw;
  height: 100%;
  padding: 16px 16px 0 16px;
  flex: 1 0 auto;
  display: flex;
  flex-direction: column;
}
.footer{
  flex: 0 0 auto;
  height: 200px;
  width: 100vw;
  margin-top: 50px;
  border-top: 1px solid #e2e2e2;
  padding: 0 72px;
  display: flex;
  align-items: center;
  justify-content: space-between;
  .footer__description{
    font-size: 16px;
    width: 40%;
  }
  a {
    font-size: 18px;
    color: #101010;
    margin-right: 10px;
    font-weight: 600;
    transition: 0.2s;
    &::after{
      height: 1px;
      background: #e2e2e2;
      width: 100%;
    }
    &:hover{
      color: #e2e2e2;
    }
    &.router-link-exact-active {
      color: #ffe033;
      text-shadow: 0 0 5px rgba(255, 224, 51, 0.13);
      // box-shadow: 0 0 10px rgba(255, 224, 51, 0.13);
    }
  }
}
</style>
```

1.8. PHP

```
<?php
header("Access-Control-Allow-Origin: *");
header('Content-Type: application/json;charset=utf-8');
header('Accept: application/json');
require 'rb-mysql.php';

R::setup('mysql:host=81.177.139.61;dbname=9295608607_pdelivery', '047245193_pdeliv', 'passtodb');

if( !R::testConnection() ){
   echo json_encode('База данных недоступна');
}
$postData = file_get_contents('php://input');
$data = json_decode($postData, true);
if(json_decode($postData, true)['apiexecute'] == null){
 $data = $_POST;
}
switch ($data['apiexecute']){
 case 1:
   if (!empty($data['phone']) && !empty($data['password'])){
     $response = R::getAll('SELECT * FROM `customer` WHERE `phone`= "'.$data['phone'].'" ');
     if(count($response) == 0){
       $response = array("errors" => "пользователь не зарегистрирован");
     }else{
       if(password_verify($data['password'], $response[0]['password'])){
         $response = array(
           "token" => $response[0]['id']
         );
       }else{
         $response = array(
           "errors" => "неверный пароль"
         );
       }
     }
     echo json_encode($response);
   }else{
     $response = array(
       "errors" => "error-get-from-db"
     );
     echo json_encode($response);
   }
   break;
 case 2:
   if (!empty($data['phone']) && !empty($data['password']) && !empty($data['name']) && !empty($data['lname']
) && !empty($data['email'])){
     $response = R::getAll('SELECT * FROM `customer` WHERE `phone`= "'.$data['phone'].'" ');
     if(count($response) == 0){
       $response = R::dispense('customer');
       $response->password = password_hash($data['password'], PASSWORD_DEFAULT);
       $response->email = $data['email'];
```

```
      $response->phone = $data['phone'];
      R::store($response);
      $response2 = R::dispense('customerdata');
      $response2->name = $data['name'];
      $response2->lname = $data['lname'];
      R::store($response2);
      $response3 = R::dispense('customercustomerdata');
      $response3->idcustomer = $response['id'];
      $response3->idcustomerdata = $response2['id'];
      R::store($response3);
      $response = array("token" => $response['id']);
     }else{
      $response = array("errors" => "пользователь уже зарегистрирован");
     }
     echo json_encode($response);
    }else{
     $response = array(
      "errors" => "error-get-from-db"
     );
     echo json_encode($response);
    }
    break;
  case 3:
    if($data['token'] != ""){
     $response = R::getAll('SELECT * FROM `order` WHERE idcustomer= '.$data['token'].' ');
     if(count($response) != 0){
      $response3 = R::getAll('SELECT product.id, name, cost, weight, imagepath, qty FROM `order` RIGHT JOI
N `product` ON order.idproduct = product.id WHERE idcustomer = '.$data['token'].' AND order.status = 0');
       if(count($response3)==0){
        $response3 = array(
         "cart"=> "0"
        );
       }
       echo json_encode($response3);

     }else{
       $response = array(
        "cart"=> "0"
       );
       echo json_encode($response);
     }
    }
    break;
  case 5:
    if($data['category']==0){
     if($data['token'] == ""){
      $response3 = R::getAll('SELECT id, name, cost, weight, imagepath FROM `product` WHERE isactive = 1 O
RDER BY id');
     }else{
```

```
      $response3 = R::getAll('SELECT product.id, name, cost, weight, imagepath, qty FROM `order` RIGHT JOI
N `product` ON order.idproduct = product.id AND idcustomer = '.$data['token'].' AND order.isactive = 1 WHERE
 product.isactive = 1 ORDER BY id');
     }
    }else{
     if($data['token'] == ""){
      $response3 = R::getAll('SELECT product.id, name, cost, weight, imagepath FROM `product` WHERE idcat
egory = '.$data['category'].' AND product.isactive = 1 ORDER BY id');
     }else{
      $response3 = R::getAll('SELECT product.id, name, cost, weight, imagepath, qty FROM `order` RIGHT JOI
N `product` ON order.idproduct = product.id AND idcustomer = '.$data['token'].' AND order.isactive = 1 WHERE
 idcategory = '.$data['category'].' AND product.isactive = 1 ORDER BY id');
     }
    }
    echo json_encode($response3);
    break;
  case 6:
    $response = R::getAll('SELECT * FROM `order` WHERE idcustomer = '.$data['token'].' AND idproduct = '.$d
ata['idproduct'].' AND isactive = 1');
    if(count($response) == 0){
     $response2 = R::dispense('order');
     $response2->idcustomer = $data['token'];
     $response2->idproduct = $data['idproduct'];
     $response2->qty = 1;
     R::store($response2);
    }else{
     $response2 = R::load('order', $response[0]['id']);
     $response2->qty = $response[0]['qty']+1;
     R::store($response2);
    }
    echo json_encode($response2);
    break;
  case 7:
    $response = R::getAll('SELECT * FROM `order` WHERE idcustomer = '.$data['token'].' AND idproduct ='.$da
ta['idproduct'].");
    if($response[0]['qty'] <= 1){
     $response2 = R::trash('order', $response[0]['id']);
     echo json_encode($response2);
    }else{
     $response2 = R::load('order', $response[0]['id']);
     $response2->qty = $response[0]['qty']-1;
     R::store($response2);
     echo json_encode($response2);
    }
    break;
  case 8:
    $response = R::getAll('SELECT * FROM `category`');
    echo json_encode($response);
    break;
  case 9:
```

```
    $response = R::getAll('SELECT email, name, lname, adress FROM `customer` LEFT JOIN `customercustomer
data` ON customer.id = customercustomerdata.idcustomer RIGHT JOIN `customerdata` on customerdata.id = cus
tomercustomerdata.idcustomerdata WHERE customer.id = '.$data['token'].");
    echo json_encode($response);
    break;
  case 10:
    $response = R::getAll('SELECT * FROM `customercustomerdata` WHERE idcustomer= '.$data['token'].' ');
    $response2 = R::load('customerdata', $response[0]['idcustomerdata']);
    $response2->name = $data['name'];
    $response2->lname = $data['lname'];
    $response2->adress = $data['adress'];
    R::store($response2);
    $response3 = R::load('customer', $response[0]['idcustomer']);
    if($data['password'] != ""){
      $response3->password = password_hash($data['password'], PASSWORD_DEFAULT);
    }
    $response3->email = $data['email'];
    R::store($response3);
    echo json_encode($response);
    break;
  case 11:
    $response = R::getAll('SELECT * FROM `customercustomerdata` WHERE idcustomer= '.$data['token'].' ');
    $response2 = R::getAll('SELECT rights FROM `customerdata` WHERE id = '.$response[0]['idcustomerdata']."
);
    echo json_encode($response2);
    break;
  case 12:
    $response = R::getAll('SELECT customer.id, email, name, lname, rights, phone, isactive FROM `customer` LE
FT JOIN `customercustomerdata` ON customer.id = customercustomerdata.idcustomer RIGHT JOIN `customerda
ta` ON customerdata.id = customercustomerdata.idcustomerdata WHERE rights = 2');
    echo json_encode($response);
    break;
  case 13:
    $response = R::getAll('SELECT customer.id, email, name, lname, rights, phone, isactive FROM `customer` LE
FT JOIN `customercustomerdata` ON customer.id = customercustomerdata.idcustomer RIGHT JOIN `customerda
ta` ON customerdata.id = customercustomerdata.idcustomerdata');
    echo json_encode($response);
    break;
  case 14:
    $response = R::getAll('SELECT * FROM `product` ORDER BY id');
    echo json_encode($response);
    break;
  case 15:
    $dir = "/img";
    $fieldname = 'image';
    $filename = $_FILES[$fieldname]['name'];
    $tmpname = $_FILES[$fieldname]['tmp_name'];
    $response2 = R::dispense('product');
    $response2->name =  $data['name'];
    $response2->cost =  $data['cost'];
    $response2->weight =  $data['weight'];
```

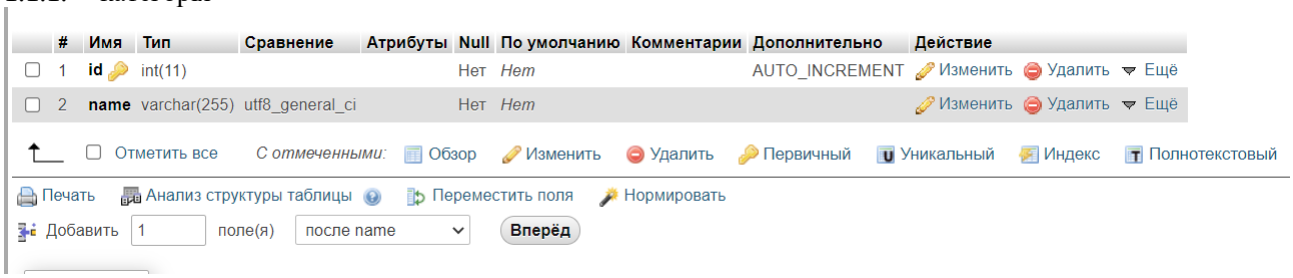
```
    $response2->qtyonstorage =  $data['qtyonstorage'];
    $response2->idcategory =  $data['category'];
    $response2->imagepath = "";
    R::store($response2);
    $id = $response2['id'];
    $response2 = R::load('product', $id);
    if(!is_dir('..'.$dir.'/'.$id.")) {
      mkdir('..'.$dir.'/'.$id.", 0777, true);
    }
    move_uploaded_file ( $tmpname , '..'.$dir.'/'.$id.'/'.$filename." );
    $response2->imagepath = ".$dir.'/'.$id.'/'.$filename.";
    R::store($response2);
    echo json_encode($response2);
    break;
  case 16:
    $response = R::load('product',$data['product']);
    $response->isactive = 0;
    R::store($response);
    echo json_encode($response);
    break;
  case 17:
    $response = R::getAll('SELECT * FROM `order` WHERE status = 1');
    if(count($response) != 0){
      $response = R::getAll('SELECT * FROM `order` WHERE idcustomer = '.$response[0]['idcustomer'].' AND  s
tatus = 1');
      for ($i=0; $i < count($response); $i++) {
        if(!empty($response[$i]['id'])){
          $response2 = R::load('order', $response[$i]['id']);
          $response2->idpostman = $data['token'];
          $response2->status = 2;
          $response2->isactive = 0;
          R::store($response2);
        }
      }
    }

    $response = R::getAll('SELECT product.name AS productname, customerdata.name, customer.phone, custome
rdata.adress, order.qty FROM `order` RIGHT JOIN `product` ON product.id=order.idproduct LEFT JOIN `custo
merdata` ON customerdata.id = order.idcustomer LEFT JOIN `customer` ON customer.id = customerdata.id WH
ERE order.status = 2 AND order.idpostman = '.$data['token'].");
    echo json_encode($response);
    break;
  case 18:
    $response = R::getAll('SELECT * FROM `customer` LEFT JOIN `order` ON customer.id = order.idcustomer
WHERE customer.id = '.$data['token'].' AND order.status = 0');
    for ($i=0; $i < count($response); $i++) {
      $response2 = R::load('order', $response[$i]['id']);
      $response2->status = 1;
      R::store($response2);
    }
    echo json_encode($response);
```

```
    break;
  case 19:
    $response = R::getAll('SELECT order.id FROM `order` WHERE order.status = 2 AND order.idpostman = '.$da
ta['token']."");
    for ($i=0; $i < count($response); $i++) {
      $response2 = R::load('order', $response[$i]['id']);
      $response2->status = 3;
      $response2->isactive = 0;
      R::store($response2);
    }
    echo json_encode($response);
    break;
}
exit;
?>
```

```
import { createStore } from 'vuex'

export default createStore({
  state: {
    userData: {
      token: "",
    }

  },
  mutations: {
    setUserData(state, userData){
      state.userData = userData
    }
  },
  getters: {
    getUserData(state){
      return state.userData
    }
  },
})

import { createApp } from 'vue'
import App from './App.vue'
import './registerServiceWorker'
import router from './router'
import store from './store'
import axios from 'axios'
import VueAxios from 'vue-axios'

createApp(App).use(VueAxios, axios).use(store).use(router).mount('#app')
```

1.9. База даних

1.1.1. категорія



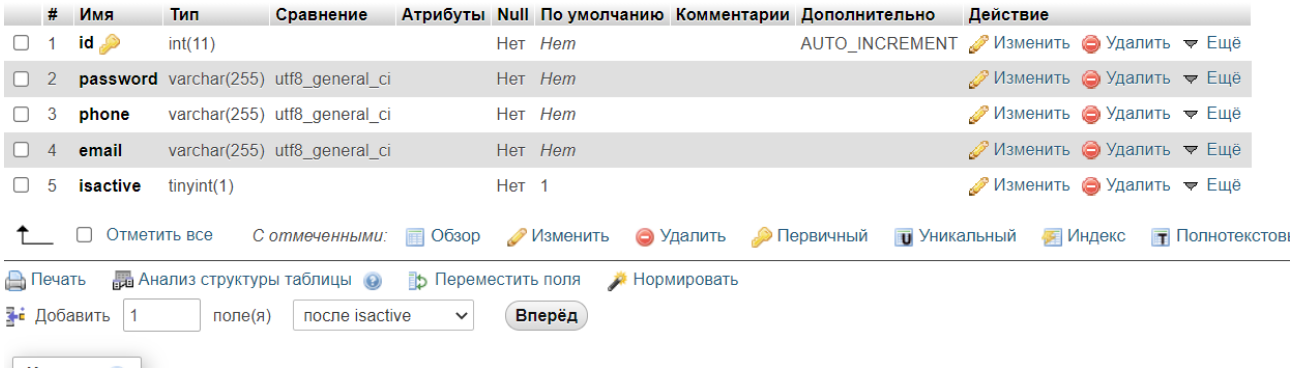
#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id	int(11)			Нет	Нем		AUTO_INCREMENT	Изменить Удалить Ещё
2	name	varchar(255)	utf8_general_ci		Нет	Нем			Изменить Удалить Ещё

Отметить все С отмеченными: Обзор Изменить Удалить Первичный Уникальный Индекс Полнотекстовый

Печать Анализ структуры таблицы Переместить поля Нормировать

Добавить 1 поле(я) после name Вперёд

1.1.2. споживач



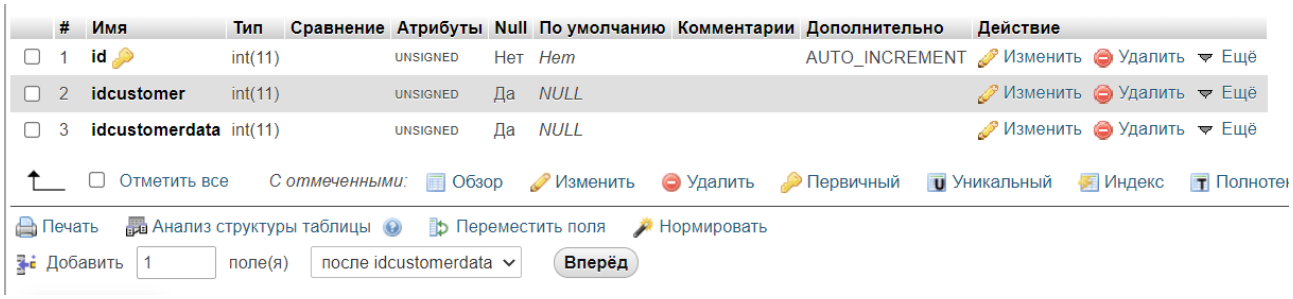
#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id	int(11)			Нет	Нем		AUTO_INCREMENT	Изменить Удалить Ещё
2	password	varchar(255)	utf8_general_ci		Нет	Нем			Изменить Удалить Ещё
3	phone	varchar(255)	utf8_general_ci		Нет	Нем			Изменить Удалить Ещё
4	email	varchar(255)	utf8_general_ci		Нет	Нем			Изменить Удалить Ещё
5	isactive	tinyint(1)			Нет	1			Изменить Удалить Ещё

Отметить все С отмеченными: Обзор Изменить Удалить Первичный Уникальный Индекс Полнотекстови

Печать Анализ структуры таблицы Переместить поля Нормировать

Добавить 1 поле(я) после isactive Вперёд

1.1.3. Клієнт



#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id	int(11)		UNSIGNED	Нет	Нем		AUTO_INCREMENT	Изменить Удалить Ещё
2	idcustomer	int(11)		UNSIGNED	Да	NULL			Изменить Удалить Ещё
3	idcustomerdata	int(11)		UNSIGNED	Да	NULL			Изменить Удалить Ещё

Отметить все С отмеченными: Обзор Изменить Удалить Первичный Уникальный Индекс Полнотек

Печать Анализ структуры таблицы Переместить поля Нормировать

Добавить 1 поле(я) после idcustomerdata Вперёд

1.1.4. Клієнт

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id	int(11)			Нет	Нем		AUTO_INCREMENT	Изменить Удалить Ещё
2	name	varchar(255)	utf8_general_ci		Нет	Нем			Изменить Удалить Ещё
3	lname	varchar(255)	utf8_general_ci		Нет	Нем			Изменить Удалить Ещё
4	rights	tinyint(1)			Нет	0			Изменить Удалить Ещё
5	adress	varchar(255)	utf8_general_ci		Нет	Нем			Изменить Удалить Ещё

Отметить все С отмеченными: Обзор Изменить Удалить Первичный Уникальный Индекс Полноте

Печать Анализ структуры таблицы Переместить поля Нормировать

Добавить 1 поле(я) после adress Вперёд

1.1.5. Замовлення

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	**id**	int(11)			Нет	*Нет*		AUTO_INCREMENT	Изменить Удалить Ещё
2	**idcustomer**	int(11)			Нет	*Нет*			Изменить Удалить Ещё
3	**idproduct**	int(11)			Нет	*Нет*			Изменить Удалить Ещё
4	**qty**	int(11)			Нет	0			Изменить Удалить Ещё
5	**status**	tinyint(4)			Нет	0			Изменить Удалить Ещё
6	**idpostman**	int(11)			Нет	*Нет*			Изменить Удалить Ещё
7	**isactive**	tinyint(1)			Нет	1			Изменить Удалить Ещё

Отметить все *С отмеченными:* Обзор Изменить Удалить Первичный Уникальный Индекс Полно

Печать Анализ структуры таблицы Переместить поля Нормировать

Добавить 1 поле(я) после isactive **Вперёд**

1.1.6. Продукти

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	**id**	int(11)			Нет	*Нет*		AUTO_INCREMENT	Изменить Удалить Ещё
2	**name**	varchar(255)	utf8_general_ci		Нет	*Нет*			Изменить Удалить Ещё
3	**cost**	varchar(255)	utf8_general_ci		Нет	*Нет*			Изменить Удалить Ещё
4	**weight**	varchar(255)	utf8_general_ci		Нет	*Нет*			Изменить Удалить Ещё
5	**imagepath**	text	utf8_general_ci		Нет				Изменить Удалить Ещё
6	**qtyonstorage**	int(11)			Нет	0			Изменить Удалить Ещё
7	**idcategory**	int(11)			Нет	0			Изменить Удалить Ещё
8	**isactive**	tinyint(1)			Нет	1			Изменить Удалить Ещё

Отметить все *С отмеченными:* Обзор Изменить Удалить Первичный Уникальный Индекс Полнотексто

Печать Анализ структуры таблицы Переместить поля Нормировать

Добавить 1 поле(я) после isactive **Вперёд**

ДОДАТОК Б

РЕЦЕНЗІЯ

на дипломний проект бакалавра

на тему:

«Розробка веб-додатку для замовлення доставки продуктів споживачам»

Студента групи 122-17-3 Олійника Данііла Валерійовича

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я Файлу	**Опис**
Пояснювальні документи	
Диплом_Олійника_Данііла.doc	Пояснювальна записка до дипломного проекту. Документ Word.
Програма	
Delivery.zip	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_Олійника_Данііла.ppt	Презентація дипломного проекту