ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, І ТЕРМІНІВ

Скорочення

AJAX – Asynchronous Javascript and XML (асинхронний JavaScript і XML);
BSON – Binary JavaScript Object Notation (формат електронного обміну цифровими даними);
CEO – Chief Executive Officer (головний виконавчий директор);
CSS – Cascading Style Sheets – каскадні таблиці стилів;
DOM – Document Object Model (програмний інтерфейс для доступу до документів);
HDD – Hard Disk Drive (жорсткий диск);
HTML – HyperText Markup Language (мова гіпертекстової розмітки);
HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту);
Haml – Hyper abstraction markup language (мова розмітки для спрощеної генерації HTML);
JS – JavaScript (язик програмування);
JSON – JavaScript Object Notation (текстовий формат обміну даними);
MIT – ліцензія Массачусетського технологічного інститута;
MVC – Model-View-Controller (модель-представлення-контролер);
MySQL – вільна реляційна система управління базами даних;
NPM – Node.js Package Manager (менеджер пакетів);
SPA – односторінкові додатки (single page application);
SVG – Scalable Vector Graphics (масштабуєма векторна графіка);
URL – Uniform Resource Locator (визначник місцезнаходження ресурсу);
БД – база даних.
ОПР – особа що приймає рішення;
ОС – операційна система;
ПК – персональний комп’ютер;
СУБД – система управління базами даних.

Терміни

JavaScript Object Notation – форма представлення даних, в який самі дані представляються у вигляді об'єкта мови JavaScript, виду «Ключ»:«значення»;
Node.js – програмна платформа, заснована на движку V8 (здійснює трансляцію JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованого мови в мову загального призначення;

TypeScript – мова програмування, представленна Microsoft в 2012 році і позиціонується як засіб розробки веб-додатків, що розширює можливості JavaScript;

Full-stack – це фахівець, який здатний брати активну участь у всіх етапах розробки веб-додатків, починаючи від серверної логіки і її реалізації за допомогою різних технологій і фреймворків, і закінчуючи клієнтським кодом, які працюють безпосередньо в браузері;

Open-source – програмне забезпечення, користувачі якого мають права («свободи») на його необмежену установку, запуск, вільне використання, вивчення, поширення і зміна (вдосконалення), а також поширення копій і результатів зміни;

Веб-додаток – клієнт-серверний додаток, в якому клієнт взаємодіє з сервером за допомогою браузера, а за сервер відповідає – веб-сервер;

ВСТУП

В даний час, тільки використовуючи якісні комп'ютерні програми, можна організувати ефективний облік процесу виробництва і реалізації хлібобулочних і кондитерських виробів.

Для підприємств, що займаються виробництвом і реалізацією хліба, дуже важлива оптимальність в використанні, простота, швидкість та зручність у роботі, при прийомі замолень від клієнтів. Також важливим завданням є створення завдань виробництва на наступну добу, розрахунок кількості інгредієнтів та хлібобулочних виробів, швидкий облік витрат сировини при виробництві хліба.

Розроблена система є тестовою, тому вона в процесі життєвого циклу буде доповнюватися та вдосконалюватися.

1 ПОСТАНОВКА ЗАВДАННЯ

В рамках даного проекту я хочу розробити систему управління пкарнею з використанням нових технологій, торкнувшись питань розподіленням ролів для користувачів, авторизації користувачів, управлінням правами доступу, також розібрати серверну та клієнтську частину додатка.

Проаналізувати існуючі веб-системи, з’ясувати які основні елементи необхідні для повноцінної роботи системи та вибрати тут технологію, котра задовольнить всі майбутні вимоги.

Будемо створювати повноцінний realtime додаток. Нам знадобитися сервер на якому буде зберігатися наша БД і запущено сам додаток.

По суті це будует робочі місце для керуючого пекарнею, а також для диспетчера. Роль керівника пекарнею буде виконувати адміністративні функції. Роль диспетчера буде виконувати прийом замовлень на наступний день, складання виробничого завдання для пекарів, прийом готової продукції на склад, розподілення готової продукції по автомобілям, друк завдання для водіїв, а також прийом повернень. Реалізовувати цей проект будемо на темі inspiniya.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Пекарня – невелике немеханізованими підприємство по випічці та реалізації хлібобулочних і кондитерських виробів, як правило, також реалізує їх на місці. Типовий асортимент пекарень складають різні хліба, торти, тістечка, каравай, кекси і пироги.

Технологічний процес виробництва хлібобулочних виробів в пекарні складається з наступних основних етапів:

- прийом і зберігання сировини;
- підготовка сировини до пуску у виробництво;
- приготування тіста;
- оброблення тіста;
- випічка виробів;
- зберігання і реалізація випечених виробів.

Організація хлібопекарні показала, альтернативність приватного підприємства – хлібозаводам, а також те, що обидві виробничі системи можуть співіснувати і змагатися на благо споживача [1].

Велика структура і організація виробничих процесів на хлібозаводах призвели до незадовільних результатів. Це характеризувалося:

- складністю у визначенні відповідального за кінцевий результат, що призвело до неможливості визначення відповідального за прийняття рішень шляху розвитку;
- відносно низькою мотивацією працюючого персоналу, що виражається в незадовільній якості праці і невисокої продуктивності;
- проблемою впровадження будь-якого нововведення, через що склалися суперечки між керівниками та працюючими, які перешкоджали прийняттю нових рішень;
- втратою індивідуальності продукції, відсутністю оригінального оформлення.

Очевидно, що ефективна діяльність, тобто ефективні витрати на виробництво продукції пристосованої до ринку, легше здійснювати в маленькій мобільного структурі, оскільки більше уваги можна приділити перевагам покупців і якості продукту [2].

Виготовлення хлібобулочної продукції – це галузь, яка при дотриманні правильної стратегії приносить високу прибутковість. Це викликано в першу чергу тим, що в сучасному суспільстві робочий день, іноді, проходить прак-

тично зі швидкістю світла. Сьогодні, людина, що працює в сфері бізнесу, просто не володіє необхідною кількістю часу, для того, щоб самостійно приготувати собі сніданок, обід або вечерю. Тому, заклади швидкого харчування стали невід'ємною частиною життя будь-якого успішного підприємця. Ні для кого не є секретом, що більшу частку раціону будь-якого підприємця займає випічка.

Якщо розглядати історичні аспекти розвитку продажів хлібобулочних виробів, то варто відзначити, що історично (15-20 років тому) склалося так, що виготовлення цієї продукції було прерогативою великих промислових заводів. Однак, пройшли роки, ситуація змінилася і виробництво хлібобулочної продукції стало можливо і на невеликих пекарнях. Ця сфера дуже приваблива і швидко окупається, тому інтерес з боку підприємців просто колосальний.

На сьогоднішній день частка кондитерських і хлібобулочних виробів вже досить значна. Наприклад, в США від всього обсягу хлібної продукції 60% виготовляють саме хлібопекарні. У Франції ця частка становить 85%, а в Німеччині – 80%. Найцікавіше, що рекорд побила все-таки невелика Іспанія, там ця частка становить аж 95% [3].

3 ОГЛЯД ІСНУЮЧИХ ТЕХНОЛОГІЙ РЕАЛІЗАЦІЇ СИСТЕМ

Нещодавно JavaScript зайняв місце серед кращих мов для вивчення за версією IBM в 2017 році. На даному етапі він використовується, як для клієнтської, так і для серверної частини і допомагає проектувати привабливі інтерфейси, збагачувати веб-додатки численними функціями і фічами, змінювати веб-сторінки в реальному часі і багато іншого [4].

Розширюючи можливості JavaScript для скриптів на стороні сервера, розробка за допомогою фреймворка Node.js вийшла на перший план серед мов програмування. Node.js привів до помітного висунення JavaScript за межі браузера, що широко підтверджується JS-розробниками.

Завдяки своїм видатним властивостям, Node.js користується високим попитом в співтоваристві бекенд-розробників і у технологічних гігантів. Основні з цих властивостей - однопотоковий подієвий цикл, асинхронний і неблокуючий процес введення / виводу. Використовуючи ці основні характеристики та підходи Node.js, існують багато фреймворків. Ці фреймворки Node.js можуть бути засобом для прискорення циклів розробки і збільшення потужності Node.js-розробки.

Переваги та сильні сторони додатків, побудованих на цих фреймворках, різні, але у них є і щось спільне. Вони всі написані на JavaScript, тому мають сумісність в плані пристроїв, операційних систем і браузерів. Ці фреймворки Node.js дозволяють вам розроблять веб-додатки в реальному часі і без сторонніх серверів додатків, веб-серверів, технологій або інструментів.

Компаніям, а також фрілансерам, які займаються розробкою на Node.Js, що знаходяться в пошуках різних стратегій для прискорення веб-розробки без втрати в якості, варто звернути увагу на ці фреймворки.

Середа JavaScript стала просто величезною. Вона має власну екосистему бібліотек, фреймворків, інструментів, менеджерів пакетів і нових мов, які компілюються до JavaScript. Цікаво, що NPM, який є де-факто менеджером пакетів для JavaScript, також є найбільшим в світі реєстром програмного забезпечення. Ось витяг з публікації, опублікованій на Linux.com ще в січні 2017 року (рис. 3.1).

З більш ніж 350 000 пакетами, реєстр NPM містить більш ніж удвічі пакетів більше у порівнянні з іншими найбільш популярними реєстрами пакетів (одним з яких є сховище Apache Maven). Фактично, в даний час це найбільший реєстр пакетів в світі.

Тепер давайте промотаємо час вперед на вісім місяців, і в даний час в реєстрі NPM є близько 500 000 пакетів. Це величезне зростання в порівнянні з іншими сховищами пакетів.

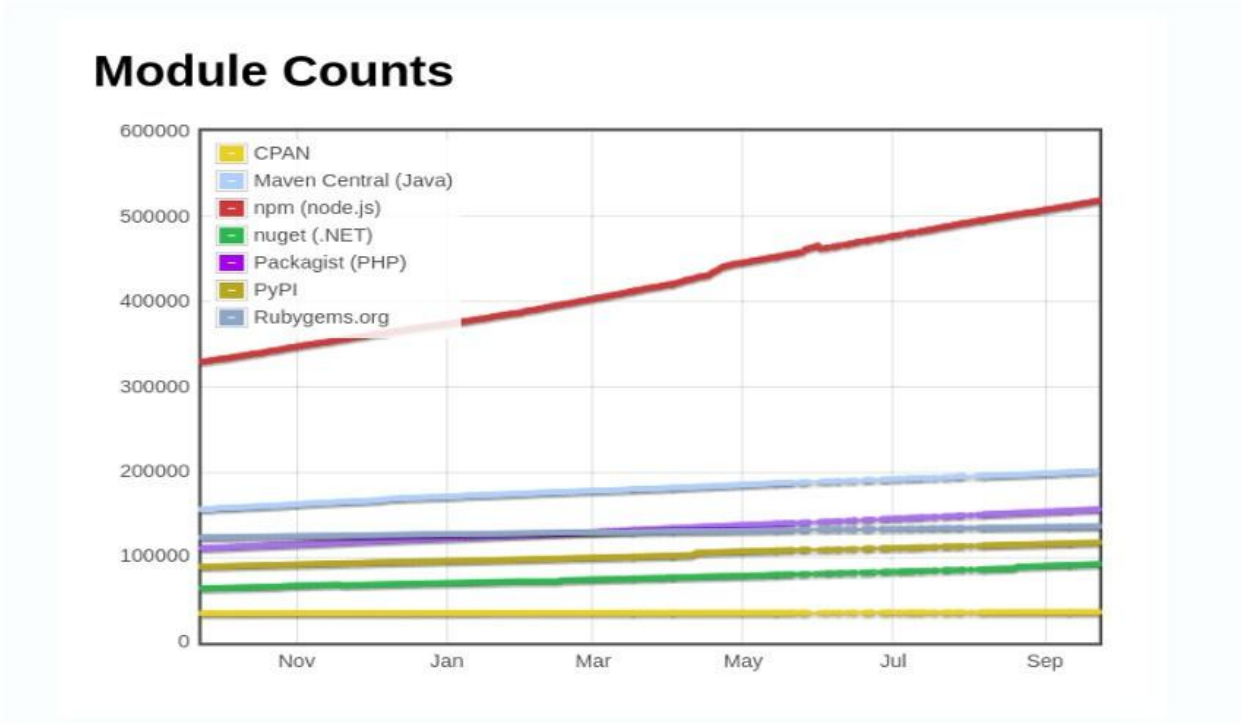


Рисунок 3.1 – Сховища пакетів

Будучи фронт-енд розробником на JavaScript, важливо не відставати від сучасних інструментів і бібліотек JavaScript. Коли технологія стає популярною, попит на неї високий, що, в свою чергу, означає більше завдань з кодування, за які платять вищу вартість в галузі.

Фреймворк має архітектуру, яка диктує потік управління в вашому додатку. Він описує основу і розповідає вам, як все повинно бути організовано. Вам також надаються основні функції, необхідні для запуску і роботи програми. Крім того, ви зобов'язані дотримуватися принципів і шаблонами фреймворка.Функціональність, така як обробка подій, створення викликів AJAX, прив'язка шаблонів та даних і тестування, вбудовані у фреймворк.

Бібліотека є багаторазовою частиною коду, яка пропонує певні функції. Це набір функцій, об'єктів і класів, які ви можете використовувати в своєму додатку. Бібліотека абстрагує різні шари, так що вам не потрібно турбуватися про їх деталі реалізації.

Ви можете викликати бібліотечну функцію і передати їй деякі параметри, бібліотека виконає її і поверне вам керування назад. Однак вона не накладає ніяких структурних обмежень, що звужують використання бібліотеки.

Різниця між фреймворком і бібліотекою полягає в тому, що ви викликаєте бібліотеку, тоді як фреймворк викликає вас. Він часто включає в себе безліч бібліотек і має більш високий рівень абстракції.

Далі описується декілька бібліотек і фреймворків, які розширюють ваш основний функціонал Node.js новітніми властивостями і будуть швидко розвиватися в 2018 році.

3.1 Бібліотеки JavaScript

3.1.1 Бібліотека React

React – це бібліотека JavaScript, створена розробниками Facebook і Instagram. React була визнана найулюбленішою технологією серед розробників, згідно з опитуванням Stack Overflow Survey 2017. React також є найпопулярнішим проектом JavaScript, грунтуючись на підрахунку зірок GitHub.

За допомогою React можна створити інтерактивний інтерфейс з використанням декларативного підходу, в якому ви можете контролювати стан додатка, кажучи «картинка повинна виглядати так». Він використовує компонентну модель, в якій компоненти є багаторазово використовуваними елементами призначеними для користувача інтерфейсу, і кожен компонент має свій власний стан. React використовує віртуальний DOM, так що вам не потрібно турбуватися про пряме маніпулювання. Інші примітні особливості React включають в себе односторонній потік даних, додатковий синтаксис JSX і інструмент командного рядка для створення проекту React з нульовою конфігурацією збірки.

Ця бібліотека є основною в інтерфейсі Фейсбук і інстаграмма, показуючи свою ефективність всередині динамічних додатків з високим трафіком (пропускною спроможністю). По праву вважається самою швидкозростаючою бібліотекою: на сьогодні налічується близько 1000 авторів Github. У MVC (Model-View-Controller) модель React.js діє як "V" і може бути легко інтегрованана в будь-яку архітектуру. На додаток до цього, компоненти React можуть бути створені і повторно використані в інших додатках або навіть передані для громадського використання.

Не дивлячись на той факт, що React більш складний у вивченні, він робить розробку додатків простою і легкою для розуміння. Крім того, він може

ідеально підійти для складних, значних програмних рішень з високим ступенем навантаження [4].

3.1.2 Бібліотека D3.js

D3 (або D3.js) – потужна бібліотека JavaScript для створення інтерактивних візуалізацій з використанням веб-стандартів, таких як SVG, HTML та CSS. У відмінності від інших бібліотек візуалізації, D3 пропонує кращий контроль над візуальним результатом.

D3 працює, зв'язуючи дані з DOM і потім перетворюючи їх в документ. Він також має власну екосистему, яка складається з плагінів та бібліотек, які розширюють її основні функціональні можливості. Бібліотека існує з 2011 року, і в ній є безліч документів і навчальних посібників, які допоможуть вам розпочати роботу.

3.1.3 Бібліотека JQuery

jQuery – це бібліотека, яка зробила JavaScript більш доступним, а маніпуляції з DOM простіше, ніж раніше. Легке навчання jQuery і простий синтаксис породили покоління нових клієнтських розробників. Кілька років тому jQuery вважався надійним рішенням для створення потужних веб-сайтів з крос-браузерної підтримкою. Основні функції jQuery, такі як маніпулювання DOM на основі селекторів CSS, обробка подій і створення викликів AJAX, підживлювали її популярність. JQuery – найпопулярніша в даний час бібліотека JavaScript, яка встановлюється на 65% з 10 мільйонів найбільш відвідуваних сайтів в Інтернеті. JQuery – це безкоштовне програмне забезпечення з відкритим вихідним кодом, ліцензоване за ліцензією MIT.

Однак все змінилося, і середовище JavaScript постійно розвивається. Деякі функції jQuery були включені в нову специфікацію ECMAScript. Більш того, нові бібліотеки і фреймворки, використовувані сьогодні, мають власний спосіб зв'язування DOM, і тому прості методи маніпуляції з DOM більше не потрібні. Популярність jQuery знаходиться на спаді, але я не думаю, що він зникне найближчим часом.

3.1.4 Бібліотека компонентів Polymer

Polymer – це бібліотека готових, вільно компонованих веб-компонентів, які можна використовувати для створення додатків або нових веб-компонентів. Polymer також включає в себе стандартні polyfill-сценарії,

які гарантують, що бібліотека буде вести себе однаково в усіх сучасних браузерах. Вони необхідні тому, що постачальники браузерів продовжують активно втілювати різні аспекти стандарту веб-компонентів, кожен в своєму темпі.

Polymer – це результат старань одного виробника браузера з підготовки свого продукту до майбутнього, де є популярними веб-компоненти. Група розробників Polymer отримує безпрецедентний доступ до ядра браузера (Chrome), тому розробка браузера тісно пов'язана з бібліотекою – що гарантує збереження бібліотекою високого рівня продуктивності і її правильне функціонування у всіх основних версіях Chrome.

При роботі з Polymer зберігається все, що ви знаєте і любите в DOM, CSS, і JavaScript. Крім того, використовуючи тільки HTML, CSS і JavaScript, тепер можна створювати багаторазові спеціальні HTML-компоненти, повністю інкапсульовані в призначений для користувача інтерфейс або функціональну частину програми.

Крім вирішення інших завдань, пов'язаних з розгортанням веб-компонентів, Polymer допомагає при реєстрації компонентів, управлінні їх життєвим циклом, обробці атрибутів, оформленні CSS-стилів, виклик методів і обробці подій. Спільнота веб-розробників поширює постійно зростаючу бібліотеку загальнодоступних компонентів з відкритим вихідним кодом. Єдине, що потрібно знати розробнику, котрий використовує веб-компоненти, створені іншими користувачами, – це новий набір HTML-тегів.

3.2 Фреймворки JavaScript

3.2.1 Фрейсворк Angular

AngularJS – JavaScript-фреймворк з відкритим вихідним кодом. Призначений для розробки односторінкових додатків. Його мета – розширення браузерних додатків на основі MVC-шаблону, а також спрощення тестування і розробки. Фреймворк працює з HTML, що містить додаткові атрибути, які описуються директивами, і пов'язує введення або виведення області сторінки з моделлю, яка представляє собою звичайні змінні JavaScript. Значення цих змінних задаються вручну або витягуються з статичних або динамічних JSON-даних [5].

AngularJS колись була найпопулярнішою технологією JavaScript серед розробників інтерфейсів. Вона була підтримана Google і співтовариством приватних осіб і корпорацій. Незважаючи на популярність, у AngularJS були

свої недоліки. Команда провела два роки роботи над новою версією Angular, яка була, нарешті, випущена у вересні 2016 року.

Після довготривалого релізу Angular його популярність досягла нових вершин, але він буде тримати планку не відступаючи і в 2017 році.

Angular.js часто називають MVW (Model-View-Whatever) фреймворк і серед основних переваг для малих і середніх компаній: швидке кодування, швидке тестування будь-якого додатку та двостороння привязка даних (зміни в базі відразу відображені на користувацькому інтерфейсі). З моменту виходу його екосистема вийшла за межі очікуванних. Сьогодні його заслужено називають найбільш використовуваними JS фреймворком для розробки односторонніх додатків (SPA Single-Page-Applications) і він може похвалитися найбільшою спільнотою розробників.

Angular2 нова версія з великим списком функцій, які дозволять розробляти все, починаючи від веб-додатків до мобільних додатків. Фреймворк побудований на TypeScript від Microsoft з прицілом на те, щоб зробити JavaScript більш гнучким та привабливим для великих підприємств.

Обидва Angular є найкращим варіантом для корпоративних додатків або для середовищ програмування з високими стандартами до читання коду [6].

3.2.2 Фреймворк Vue.js

Vue.js – це легка інфраструктура JavaScript, яка в цьому році була в тренді. Це найпопулярніший фреймворк JavaScript на GitHub з точки зору зірок GitHub. Синтаксис шаблону на основі HTML пов'язує наданий DOM з даними екземпляру.

Фреймворк пропонує досвід, схожий на React, з його віртуальними DOM і компонентами багаторазового використання, які можна використовувати для створення як віджетів, так і цілих веб-додатків. Крім того, ви також можете використовувати синтаксис JSX для безпосереднього написання функцій рендеринга. Коли стан змінюється, Vue.js використовує систему реактивності, щоб визначити, що змінилося і перерозподіляє мінімальну кількість компонентів. Vue.js також підтримує інтеграцію інших бібліотек у фремворк без особливого клопоту.

Vue.js кращий вибір для швидкої розробки крос-платформних додатків. Він може стати міцною основою для високопродуктивних додатків в одну сторінку (SPAs) і вигідним рішенням для тих випадків, коли продуктивність важливіше, ніж хороша організація коду або структура програми.

3.2.3 Фреймворк Meteor.js

Meteor один з найпопулярніших JS фрейворків, який стійко крокує з великою кількістю функцій для бекенда розробки та відтворення фронт-енду, управління базами даних і бізнес логікою. З випуску в 2012 році його екосистема зросла кардинально швидкими темпами. Ця full-stack платформа дозволяє швидко розробляти веб і мобільні додатки. З точки зору продуктивності, всі зміни в базі даних відразу передаються на призначений для користувача інтерфейс і назад без будь-яких видимих втрат часу, викликаних різницею в мовах або в часі реакції сервера.

Meteor.js охоплює всі етапи циклу розробки програмного забезпечення і піклується про такі процеси, як лінкінг, конкатенація файлів та інше. Фреймморк зараз використовується для розробки додатків реального часу в таких компаніях, як Mazda, IKEA, Honeywell і багатьох інших [4].

Meteor – це MVC Node.js фреймворк з відкритим вихідним кодом і великими можливостями в плані розробки мобільних і веб-додатків. Цей фреймворк дозволяє вам будувати додаток як для клієнтської так і для серверної сторони в реальному часі. Meteor головним чином підтримує Linux, Windows і macOS.

Використовуючи реактивну модель програмування, пропоновану Meteor, ви можете створювати додаток з меншою кількістю коду JavaScript. За допомогою Meteor створені деякі популярні додатки, такі як пошук роботи Blonk або додаток для спільної командної роботи Respondly.

У Meteor є протокол DDP, який дозволяє вам підключатися до чого завгодно на стороні сервера, наприклад до сховища даних підприємства або до простої базі даних, або до датчиків інтернету речей. Meteor може швидко інтегруватися з MongoDB.

Meteor користується хорошим попитом на ринку: у нього більше 28 тис. Зірок на GitHub і величезне співтовариство підтримки. Абсолютно все в Meteor працює з коробки.

3.3 Інструменти розробки

На відміну від бібліотек та фреймворків, інструмент зазвичай виконує завдання на клієнтському коді. Він бере ваш код в якості вхідних даних, виконує на ньому завдання, а потім повертає результат. Зазвичай використовуються інструменти включающі в себе менеджери файлів і інструменти для збірки, активатори, модулі та інструменти для скаффолдингу.

Інструменти: універсальні виконавці завдань. Універсальні виконавці завдання – це інструменти, які використовуються для автоматизації певних завдань, що повторюються.

3.3.1 Інструмент збірки веб-додатка – Gulp

Gulp – це інструмент збірки веб-додатки, що дозволяє автоматизувати повторювані завдання, такі як складання і мініфікація CSS- і JS-файлів, запуск тестів, перезавантаження браузера і т.д. Тим самим Gulp прискорює і оптимізує процес веб-розробки. Інструментарій JavaScript, який використовується як виконавець завдань і як система збирання в веб-розробці. Gulp спрощує процес написання завдань навіть для людей, які менш знайомі з JavaScript.

Gulp використовує конвеєри для потокової передачі даних з одного плагіна в інший, і кінцевий результат виводиться в папку призначення [7].

3.3.2 Повнофункціональний текстовий редактор – Atom

Творці відомого сайту для програмістів Github вирішили допомогти своїй цільовій аудиторії і створити щось, що розробники можуть використовувати кожен день. Так з'явилася ідея текстового редактора Atom.

Головна особливість Atom – багаті можливості по налаштуванню. Редактор можна налаштувати на свій смак. Спочатку в нього вбудовані файл-менеджер, просунуті функції пошуку і заміни, різноманітні курсори, опції згортання коду, ясний інтерфейс, можливість імпорту правил.

Програма Atom має повний доступ до файлової системи, нативні для операційної системи меню та панель команд. При цьому вона ідеально пристосована для веб-програмування: можна додавати власні функції для редагування CSS, HTML і JavaScript.

Потрібно відзначити також інтеграцію з Node.js, включаючи запуск веб-сервера прямо з редактора. Будь-яка з 50 тис. Бібліотек для Node.js викликається з редактора.

Atom включає в себе більше 50 open-source пакетів, які працюють поверх мінімального ядра. Максимально адаптивну систему створювали з думкою про те, щоб «стерти межу між розробником і користувачем», як сказано на офіційному сайті. Це означає, що архітектура проста і зрозуміла для кожного: можна замінити який-небудь пакет своїм власним – і закачати його в центральний репозиторій, щоб їм скористався кожен охочий [8].

3.4 Висновки з обзору технологій

JavaScript залишився актуальним з часів його створення ще в 1995 році. Ймовірно, він залишиться таким же, поки браузери не вирішать відмовитися від нього заради іншої мови. Незважаючи на те, що існує безліч інших мов, які компілюються з використанням JavaScript, немає іншої мови сценаріїв, який замінить JavaScript в доступному для огляду майбутньому. JavaScript став занадто популярним, щоб його можна було замінити.

Мова проста в вивченні і є безліч фреймворків і бібліотек, які допоможуть вам зайнятися його вивченням. Якщо ви шукаєте додаткові ресурси для вивчення або використання мови в своїй роботі, то в Інтернеті ви зможете знайти масу корисних ресурсів на цю тему.

Середа JavaScript швидко розвивається, що видно з поточних тенденцій в веб-розробці. Старі бібліотеки і фреймворки були замінені новими технологіями. jQuery, яка колись була найбільш популярною бібліотекою JavaScript, відчуває певний спад з точки зору привабливості, використання і популярності. Нове покоління інтерфейсних бібліотек, фреймворків і інструментів набирає силу і отримує загальне визнання.

Звикання до нових тенденцій в технології також має свої переваги. Роботи з кодування, що вимагають знання React, мають одні з найвищих зарплат в даній галузі. Тому потрібно продовжити навчання і експериментувати з новітніми інструментами і фреймворками, щоб максимально використовувати JavaScript.

4 ВИБІР І ОБГРУНТУВАННЯ ТЕХНОЛОГІЇ ГРУНТУЮЧИСЬ НА МЕТОДІ ГОЛОВНОГО КРИТЕРІЮ

Задача вибору і обґрунтування технології, на базі якої буде розроблена система, відноситься до завдань прийняття рішень в умовах багатокритеріальності.

Багатокритеріальна задача вибору формулюється в наступному вигляді. Дано безліч допустимих альтернатив, кожна з яких оцінюється безліччю критеріїв. Потрібно визначити найкращу альтернативу. При її вирішенні основна складність полягає в неоднозначності вибору найкращого рішення. Для її усунення використовуються дві групи методів. У методах першої групи прагнуть скоротити число критеріїв, для чого вводять додаткові припущення, що відносяться до процедури ранжирування критеріїв і порівняння альтернатив. У методах другої групи прагнуть скоротити число альтернатив у вихідній множині, виключивши свідомо погані альтернативи [9].

До методів першої групи відносяться метод згортки, метод головного критерію, метод граничних критеріїв, метод відстані. Слід зазначити, що суворе обгрунтування цих методів відсутнє і їх застосування визначається умовами завдання і перевагою особи яка приймає рішення.

Завдання багатокритеріальної оптимізації розглянемо в наступній постановці. Нехай задано безліч можливих (допустимих) рішень, позначимо його через D і безліч критеріїв їх оцінки (критеріїв) $f_1(x), f_2(x) \dots f_m(x)$, визначених на всіх елементах множини. Потрібно знайти рішення $x \in D$, для якого значення окремих критеріїв максимально можливі.

Зовні це завдання нагадує гру m осіб (гравців), в якій множина D – це безліч випадків в грі, а функції $f_i(x), i = \overline{1, m}$ – це функції виграшу гравців.

Однак є різниця, яка полягає в тому, що в грі гравці прагнуть окремо, індивідуально максимізувати свій виграш за рахунок вибору своєї стратегії, а в задачі багатокритеріальної оптимізації вибрати найкращий результат який необхідний особі, що приймає рішення (ОПР), тобто в ігровому завданні кожен гравець прагне досягти свого максимального результата, а в багатокритеріальному завданні рішення приймає одна людина, яка приймає рішення.

Розглянемо методи розв'язання задачі, які часто використовувані на практиці.

4.1 Метод «ідеальної» точки

Знайдемо максимально можливі значення окремих критеріїв, тобто для кожного вирішимо завдання максимізації функції $f_i(x)$ на множині D . Оптимальні значення позначимо

$$w_i = max_{x \in D} f_i(x), i = \overline{1,m} \tag{4.1}$$

Таким чином, w_i є максимально можливим значенням i -го критерію $f_i(x)$ на множині D . Точка $x_i \in D$, така, що $w_i = f_i(x_i) = w_i$, є рішенням звичайної задачі однокритеріальної оптимізації.

Покладемо $w^* = (w_1, w_2, \dots, w_m)$. Точка w називається ідеальною, оскільки в ній всі критерії мають максимально можливі значення і отримати більші значення ні одним критерієм без зменшення значень інших критеріїв, неможливо. Потім вирішуємо завдання відшукання такого вектора w , для якого відстань до ідеальної точки $p(w^*, w)$ мінімальна. В якості функції відстані $p(x, y)$ можна брати різні метрики, зокрема, середньоквадратичне відхилення

$$p(x, y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \tag{4.2}$$

або в даному випадку

$$p(w^*, w) = \sqrt{\sum_{i=1}^{m} (w_i^* - w_i)^2} \tag{4.3}$$

Таким чином, знаходиться найбільш близька до ідеальної точці точка на множені

$$W = \{(f_1(x), f_2(x) \dots f_m(x)), x \in D\} \tag{4.4}$$

Вона і оголошується рішенням завдання багатокритеріальної оптимізації [9].

4.2 Лексикографічна оптимізація

Нехай окремі критерії різняться за значимістю. Впорядкуємо їх так, щоб найбільш значущим був критерій $f_1(x)$, наступним за значимістю $f_2(x)$, і т.д.

Спочатку вирішується завдання максимізації найважливішим критерієм $f_1(x)$ на всій множині допустимих рішень D . Позначимо максимальне значення в цьому завданні

$$\overline{\mu_1} = max_{x \in D} f_1(x) \tag{4.5}$$

а через D_1 – множина оптимальних рішень цього завдання:

$$D_1 = \{x \in D : f_1(x) \geq \overline{\mu_1}\} \tag{4.6}$$

тобто множина всіх $x \in D$, на яких досягається максимальне значення першого критерія. На наступному кроці вирішується завдання максимізації наступного за важності другого критерію $f_2(x)$, на множині оптимальних рішень по першому критерію. Позначимо оптимальне значення в цьому завданні через

$$\overline{\mu_2} = max_{x \in D_1} f_2(x) \tag{4.7}$$

а множина оптимальних рішень через D_2:

$$D_2 = \{x \in D_1 : f_2(x) \geq \overline{\mu_2}\} \tag{4.8}$$

Далі вирішується задача максимізації наступного за важливістю критерія $f_2(x)$ на множині оптимальних рішень попередньої задачі і так далі аж до завдання максимізації останнього критерію $f_m(x)$. . Таким чином, кожна наступна задача максимізації чергового окремого критерію вирішується на множині оптимальних рішень попередньої задачі.

Відзначимо неприємну особливість розглянутого методу, яка полягає у тому, що якщо на якомусь етапі рішення відповідної задачі виявилося єдиним, то всі наступні кроки стають недоречними, а решта критеріїв фактично не беруть участі і не враховуються при визначенні рішення вихідної задачі.

4.3 Метод згортки критеріїв

Основна ідея методу згортки приватних критеріїв полягає в тому, щоб сформувати єдиний інтегральний критерій якості рішення, враховуючий і агрегуючий в собі всі окремі критерії і, тим самим, звести рішення задачі багатокритеріальної оптимізації до вирішення завдання максимізації цього інтегрального критерію.

У загальному вигляді метод згортки окремих критеріїв можна уявити наступним чином. Здається функція згортки $\varphi(u_1, u_2, \ldots, u_m)$. Кількість змінних у функції згортки дорівнює числу окремих критеріїв. Єдиний інтегральний критерій у вихідній задачі формується як суперпозиція функцій окремих критеріїв і функції згортки:

$$\Phi(x) = \varphi(f_1(x), f_2(x), \ldots, f_m(x)) \tag{4.9}$$

Тим самим вихідна багатокритеріальна задача зводиться до вирішення задачі максимізації інтегрального критерію $\Phi(x)$ на множині D .

Задаючі різні функції згортки, можна отримувати різні конкретні інтегральні критерії якості. Розглянемо деякі найбільш поширені способи згортання окремих критеріїв [10].

Нехай $\varphi(u) = \sum_{i=1}^{m} \alpha_i(u_i - \beta_i), \alpha_i > 0, \beta_i \geq 0, i = \overline{1,m}$,. Тоді єдиний критерій ефективності представляється у вигляді лінійної комбінації окремих крітеріїв наступного вигляду $\varphi(u) = \sum_{i=1}^{m} \alpha_i(f_i(x) - \beta_i)$ і вирішується завдання максимізації цього критерію на множині D .

Цей спосіб зазвичай називають економічним способом згортання, так як вагові коефіцієнти α_i, β_i часто виступають в ролі цін на i -й товар, і критерій $\Phi(x)$ представляє собою сумарний прибуток. Призначення конкретних значень коефіцієнтів α_i, β_i викликає чималі труднощі.

Якщо функцію згортки взяти у вигляд і $\varphi(u) = min_{i=\overline{1,m}} \alpha_i(u_i - \beta_i)$, то інтегральний критерій буде мати вигляд

$$\Phi(x) = min_{i=\overline{1,m}} \alpha_i(f_i(x) - \beta_i) \tag{4.10}$$

Максимізація цього критерію може сприяти підтягування відстаючих критеріїв до рівня інших (фініш за останнім).

4.4 Спосіб формування мети якісного типу

Під якісними цілями розуміються цілі, які можуть бути або досягнуті, або не досягнуті (часткове досягнення мети неможливо). Усі дії, що призводять до досягнення мети, однаково хороші, точно такі ж результати, що не приводять до досягнення мети, однаково незадовільні.

Критерій ефективності приймає в цьому випадку два значення, наприклад, 1 у разі успіху і 0 в іншому випадку. Розбиваємо множину значень окремих критеріїв на задовільні і незадовільні. Назначаються деякі числа $\overline{\delta_i}, i = \overline{1,m}$, і задовільними значеннями оголошуються тільки такі, для яких $f_i(x) \geq \overline{\delta_i}\ i = \overline{1,m}$. При цьому критерій-згортка приймає наступний вигляд:

$$V(x) = 1 \text{ при } f_i(x) \geq \overline{\delta_i}, i = \overline{1,m} \tag{4.11}$$

$$V(x) = 0 \text{ в інших випадках.} \tag{4.12}$$

Цей спосіб утворення єдиного критерію найбільш доступний розумінню оперуючої сторони, так як найближче відображає сенс вимог, накладаємих на значення окремих критеріїв. Однак труднощі коректного призначення порогових значень $\overline{\delta_i}$ так, щоб не втратити найбільш ефективні способи дій і не потрапити в область недосяжних значень, робити цю процедуру є не менше суперечливо і неоднозначно, ніж спрямованість до одночасного збільшення значень всіх окремих критеріїв [10].

4.5 Метод головного критерія

У методі головного критерію в якості цільовой функції вибирається один з функціоналів f_i, наприклад f_1, найбільш повно з точки зору дослідника відображає мету прийняття рішення. Інші вимоги до результату, описувані функціоналами $f_1, \ldots, f_m$, враховуючи введення необхідних додаткових обмежень. Таким чином вирішується однокритеріальних завдання виду

$$f_1(x) \to max_{x \in D}; D' \subseteq D \subseteq R^n \tag{4.13}$$

$$D' = \left\{ x \in \frac{D}{f_i(x)} \geq t_i, i = 2, \ldots, m \right\} \tag{4.14}$$

Формально отримали більш просту задачу пошуку максимуму функціонала f_1, на новій допустимій множині D'. Додалися обмеження виду $f_i(x) \geq t_i$ яке показує, що ми згодні не намагаємося отримати макимальних значень для функціоналів $f_2, \ldots, f_m$ зберігаючи вимога їх обмеженості знизу на прийнятних рівнях. Також відмітимо, що застосування цього методу на інтуїтивному рівні зазвичай стикається на труднощі, пов'язані з можливою наявністю декількох «головних» критеріїв, які перебувають в протиріччі один з одним. Крім того, не завжди зрозумілий алгоритм вибору нижніх меж t_i. Їх необґрунтоване завдання може привести, зокрема, до порожньої множини D' [10].

Після аналізу існуючих методів, які використовуються для прийняття рішення в умовах багатокритеріальності, мною був вибраний метод головного критерію, адже вважаю що вибір із існуючих альтернатив потрібно обгрунтовувати головним критерієм, який є найважливішим з точки зори особи яка приймає рішення.

4.6 Формуліровка завдання вибору архітектури

Здійснити вибір веб-платформи для системи управління пекарнею за сукупністю декількох показників з урахуванням обмежень, що відображають індивідуальні переваги, який виступає в якості особи, яка приймає рішення (ОПР), а також можливості той чи іншой веб-платформи.

4.6.1 Формування вихідних даних

Таблиця вихідних варіантів. Використовуючи веб-сторінки, літературу, технічну документацію сформувати в табличному процесорі Excel таблицю даних про різні фреймворки. В результаті поверхнього аналізу отримаю декілька веб-платформ, серед яких буде обраний необхідний.

Таблиця повинна включати один стовпець зі списком з декількох варіантів різних фреймворків і n стовпців їх різнорідних показників, наприклад, таких: «Складність проекту», «Ефективність», «Безпека», «Витрати». Ці показники часто називаються окремими критеріями вибору. У таблиці ці критерії повинні бути пронумеровані.

Слід відразу ж передбачити два додаткових стовпчика – з назвами «Номер варіанта» в лівій частині таблиці і «Ознака видалення» в правій частині.

Обмеження. Під стовпцем кожного показника необхідно розташувати обмеження, що відображають переваги і можливості покупця.

Обмеження може мати одну з наступних форм:

- обмеження зверху, у вигляді: <= ; це обмеження означає: «не більше 200000»;
- обмеження знизу, у вигляді: >=24; це обмеження означає: «не менше 24»;
- інтервальний обмеження, у вигляді: [1;5]; це обмеження означає: «Від 1 до 5 включно»; якщо одна або обидві границі в допустимий інтервал не входять, то відповідна дужка замінюється на круглу, (1;5] наприклад, означає «Більше 1, але не більше 5»;
- перелік допустимих варіантів, у вигляді: {Червоний, Білий, Чёрний};
- перерахування неприпустимих варіантів, у вигляді: НЕ {Україна, Китай};
- відсутність обмежень – у вигляді: *;

Приклад таблиці вихідних варіантів див. табл. 4.1. В процесі виконання завдання повинна бути складена інша аналогічна таблиця, в якій осередки останнього стовпця заповнюються на другому етапі рішення, на поточному – першому етапі – осередки цього стовпця залишаються порожніми.

Таблиця 4.1 – Вхідні варіанти

Номер варіанта	Найменування js-фреймворків	Складність проекту	Ефективність	Безпека	Витрати
1	Meteor.js	3	4	4	4
2	Angular.js	4	3	3	4
3	React.js	4	2	5	5
Обмеження	*	[3;5]	>=3	[2;5]	*

4.6.2 Формування множини допустимих варіантів

Таблиця допустимих варіантів формується шляхом видалення з таблиці вихідних варіантів тих рядків, у яких хоча б один показник (окремий критерій) не задовільняє умовам, які встановленні обмеженням.

На другому етапі вирішення поставленого завдання необхідно виконати наступні дії:

1) в таблиці вихідних варіантів переглядаємо рядки кожного варіанту і порівнюємо значення кожного окремого критерію зі значенням відповідного обмеження;

2) якщо значення якогось окремого критерію не задовольняє обмеження, то в осередку «Ознака видалення» поточного рядка розміщуємо номер цього окремого критерію; якщо зустрінуться ще окремі критерії, що не відповідають відповідним обмеженням, то їх номери так само поміщаються в осередок стовпця «Ознака видалення». Всі ці номери є вказівками на необхідність виключення поточного варіанту з подальшого розгляду;

Далі:

1. Якщо таблиця допустимих варіантів виявиться порожньою, то необхідно повернутися на етап формування таблиці вихідних варіантів.
2. Якщо таблиця допустимих варіантів складається з одного рядка, то рішення задачі отримано.
3. Якщо таблиця допустимих варіантів складається з декількох рядків, то здійснюється перехід до наступного етапу рішення.

Таблиця допустимих варіантів для нашого прикладу див. табл. 4.2 не порожня, не складається з одного елемента, отже, необхідно виконувати наступний етап рішення.

Таблиця 4.2 – Допустимі варіанти

№	Найменування js-фреймворків	Складність проекту	Ефективність	Безпека	Витрати	Ознака видалення
1	Meteor.js	3	4	4	4	
2	Angular.js	4	3	3	4	
3	React.js	4	2	5	5	2

4.6.3 Ранжування вербальних (словесних) значень

Деякі параметри варіантів мають числові значення, а деякі – словесні. Словесні значення називаються якісними, як протилежність кількісним (числовим) значенням. Поряд з терміном «словесні» використовується терміни «вербальні» і «лінгвістичні». Еквівалентні за змістом словосполучення «словесні значення», «вербальні значення» і «лінгвістичні мітки».

Щоб порівнювати словесні значення між собою, для них необхідно встановити ранги – їх номери по спадаючій суб'єктивної переваги. По суті справи, встановлення рангів – це визначення переваги на совокупності слів,

що утворюють безліч значень вербального показника. Ранг – це аналог сортності товару або зайнятого місця в змаганні. Вищий ранг дорівнює 1, далі йдуть ранги 2, 3, і так далі до числа порівнюваних слів n. Чим вище ранг, тобто чим менше його числове значення, тим краще («краще», «якісніше») вербальне значення з цим рангом.

Процедура встановлення рангів називається ранжируванням, або упорядкуванням. Ранжування можна розглядати як відображення множини слів в множині натуральних (цілих позитивних) чисел. Іноді, обговорюється застереження щодо вимоги, щоб відображення слів в рангах було однозначним, тобто рівні ранги для різних слів не допускаються, але в багатьох додатках, в тому числі в розглянутому методі головного критерію, ця вимога не обов'язкова, і рівні ранги цілком припустимі. Більш того рівність рангів відображає цілком реальну ситуацію рівності переваг словесних значень.

Відношення переваги являє собою змістовну окрему інтерпретацію математичних відносин порядку. При цьому, якщо рівність рангів допускається, то має місце відношення несуворого порядку, в іншому випадку – відношення строгого порядку.

Ранги встановлюються за допомогою таблиці парних порівнянь, що заповнюється балами переваг, таким чином:

1) рядки і стовпці матриці маркеруються вербальними значеннями (словами) або їх скороченнями;
2) в діагональних осередках розташовуються нулі – ознаки еквівалентності значень;
3) в кожному осередку розміщується «1» (одиниця), якщо на думку ОПР слово, в рядку, краще за слово, в стовпці; в іншому випадку, якщо слово, поступається за якістю речі, осередок заповнюється «-1» (одиницею зі знаком мінус);
4) кожний осередок заповнюється значенням, протилежним значенням осередка, симетричного відносно діагоналі, тобто в ній поміщається «-1», якщо симетричний осередок містить «1» і поміщається «1», якщо симетричний осередок містить «-1»;
5) для кожного рядка обчислюється сума значень комірок (балів);
6) для кожного рядка встановлюється ранг за сумою балів: чим більша сума, тим менше ранг.

Авжеш, при цьому може скластися ситуація, в якій деякі вербальні значення наберуть однакову кількість балів. В цьому випадку, звичайно ж, можна вирішити задачу повторно, окремо для цих значень, але в контексті даної

задачі доцільно всім значенням з однаковими сумами балів призначити однакові ранги.

Звичайно ж, ранжувати потрібно тільки ті словесні значення, які представлені в таблиці допустимих варіантів. Ранжувати всі значення, що містяться в таблиці вихідних варіантів, сенсу немає.

4.6.4 Формування множини Парето

Таблиця варіантів, що утворюють множину Парето, формується шляхом видалення з таблиці допустимих варіантів домінуючих варіантів.

Відношення домінування позначається так: $B_i \succ B_j$. Кажуть, що варіант B_i домінує варіант B_j, і відповідно варіант B_j домінується варіантом B_i, якщо одночасно справедливі наступні дві умови:

1) серед показників якості варіанту B_i немає жодного показника, який був би гірше за відповідний показник варіанту B_j;
2) хоча б один з показників якості варіанту B_i краще за відповідний показник варіанту B_j.

Таким чином, домінуючий і домінований варіанти можуть мати рівні показники якості, але хоча б по одному показнику перший краще другого.

Таблиця множини Парето не повинна містити домінованих варіантів. Для її отримання необхідно порівняти всі рядки один з одним. Рекомендується наступний алгоритм:

1) перший рядок порівнюється з другим, третім і так далі; якщо в процесі порівняння зустрінеться домінований рядок, то він отримує ознаку видалення (у відповідне поле рядка записується номер домінуючого варіанту), якщо зустрінеться домінуючий рядок, то ознака видалення отримує перший рядок і здійснюється перехід на наступний пункт алгоритму;
2) другий рядок порівнюється з третім, четвертим і так далі; якщо в процесі порівняння зустрінеться домінований рядок, то він отримує ознаку видалення; якщо зустрінеться домінуючий рядок, то ознака видалення отримує другий рядок і здійснюється перехід на наступний пункт алгоритму і так далі;
3) передостанній рядок порівнюється з останнім рядком; якщо останній рядок виявиться домінованим, то він отримує ознаку видалення; якщо ж останній рядок виявиться домінуючим, то ознаку видалення отримує передостанній рядок;

4) всі рядки, які отримали ознаку видалення, фізично видаляються з таблиці.

Якщо таблиця множини Парето складається з одного рядка, то рішення задачі отримано.

Якщо таблиця множини Парето складається з декількох рядків, то здійснюється перехід до наступного етапу рішення.

В нашому випадку таблиця множини Парето не порожня, не складається з одного елемента, отже, необхідно виконувати наступний етап рішення.

4.6.5 Встановлення рангів показниками (критеріями)

Для вибору «кращого» варіанту за методом головного критерію необхідно встановити ранги показників (критеріїв) – їх номери по зменшенням важливості згідно з суб'єктивної думки ОПР.

Сукупність показників (критеріїв) – це, по суті справи, те ж безліч вербальних значень, тому ранги критеріїв встановлюються за допомогою вже розглянутої раніше (див. п. 4.7.3) процедури, заснованої на використанні таблиці парних порівнянь, що заповнюється балами переваг.

Для розглянутого прикладу можливий варіант таблиці парних порівнянь окремих критеріїв представлений в таблиці 4.3. Результати ранжування представлені в таблиці 4.4. Розглядаючи попередні дві таблиці, можна помітити наступне:

1) окремий критерій «Ефективність» виявився найбільш важливим для ОПР; цей показник якості набрав максимально можливу суму балів, оскільки переміг у всіх парних порівняннях;
2) окремий критерій «Складність проекту» виявився не дуже важливим для ОПР; цей показник якості набрав мінімально можливу суму балів, оскільки програв у всіх парних порівняннях; якби такий стан справ був би відомим заздалегідь, то цей показник можна було б виключити з розгляду зовсім;
3) суми балів окремих критеріїв рівномірно зменшуються від 3 до -3.

Таблиця 4.3 – Визначення рангів окремих критеріїв

Окремий критерій	Складність проекту	Ефектив-ність	Безпека	Витрати	Сума балів	Ранг
Складність проекту	0	-1	-1	-1	-3	4

Ефектив-ність	1	0	1	1	3	1
Безпека	1	-1	0	1	1	2
Витрати	1	-1	-1	0	-1	3

Таблиця 4.4 – Ранги окремих критеріїв

Окремий критерій	Ефективність	Безпека	Витрати	Складність проекту
Ранг	1	2	3	4
Сума балів	3	1	-1	-3

4.6.6 Вибір варіанту за методом головного критерію

Кращий варіант вибирається з таблиці варіантів множини Парето наступним чином:

1) вибирається найкращий варіант за головним критерієм – з найвищим рангом (пам'ятаємо, що найвищий ранг має найменший номер займаного місця, як у учасників спортивних змагань);
2) якщо знаходиться єдиний найкращий варіант, то рішення задачі отримано;
3) якщо з'ясовується, що кілька варіантів є еквівалентними за головним критерієм, то здійснюється перехід до розгляду варіантів по головному з інших критеріїв – наступного за встановленими рангах;
4) якщо всі варіанти виявляються еквівалентними за всіма критеріями, то кращий варіант вибирається за жеребом.

У моєму випадку завдання вибору варіанта головним критерієм виявився показник якості «Ефективність». Найкращим і єдиним варіантом за цим критерієм є варіант «Meteor.js», що розміщувався під номером 1 в таблиці вихідних варіантів.

4.6.7 Аналіз варіантів: додаткове обгрунтування прийнятого рішення

Очевидним є факт: для кожного з розглянутих альтернативних варіантів можна вказати місце, яке цей варіант займає по кожному з приватних критеріїв. Розглянемо варіанти множини Парето, розташувавши їх в порядку убування якості головного критерію і перерахувавши критерії в порядку убування їх рангів, що виражають значимість (важливість) цих критеріїв для ОПР див. табл. 4.5.

Потім в таблиці 4.5 кожне значення кожного окремого критерію замінимо на номер місця, яке дане значення займає в рамках даного критерію. В результаті отримаємо таблицю 4.6

Таблиця 4.5 – Кращий варіант і його конкуренти

№	Найменування js-фреймворків	Ефектив-ність	Безпека	Витрати	Складність проекту
		1	2	3	4
1	Meteor.js	4	4	4	3
2	Angular.js	3	3	4	4

Таблиця 4.6 – Порівняння варіантів по місцях і сумою місць

№	Найменування js-фреймворків	Ефек-тив-ність	Безпека	Витра-ти	Склад-ність проек-ту	Сума місць	
		1	2	3	4	Перших двух	Зага-льна
1	Meteor.js	1	1	1	2	2	5
2	Angular.js	2	2	1	1	4	6

Додаткове обґрунтування прийнятого рішення (обраного варіанту) здійснимо шляхом визначення показника, який назвемо рівнем якості і позначимо літерою Θ.

Визначення: рівень якості кращого за головним критерієм варіанти – це лінгвістична мітка, значення якої обчислюється за формулою:

(4.15)

$$\Theta = \begin{cases} \text{Дуже добре, якщо } 0{,}3 \le \frac{\xi}{n} < 0{,}4 \\ \text{Відмінно, якщо } 0{,}4 \le \frac{\xi}{n} < 1 \end{cases} \quad (4.16)$$

де n – загальна кількість окремих критеріїв;

ξ – кількість перших (у ранзі) окремих критеріїв, за сумою місць в яких кращий за головним критерієм варіант зберігає лідируюче положення.

Значення ξ, знаходиться шляхом обчислення для кожного варіанта суми місць за двома, трьома і так далі першим критеріям – до тих пір, поки найкращий варіант буде зберігати лідируюче положення.

У нашому випадку кращий за головним критерієм варіант № 1 зберігає лідируюче положення по сумі місць, займаних за двома першими критеріями, тобто в нашому прикладі $\xi = 2$. Оскільки загальна кількість врахованих окремих критеріїв дорівнює $n = 4$, то маємо відношення $\frac{\xi}{n} = 0.5$. Отримане значення відношення $\frac{\xi}{n}$ згідно з визначенням означає, що рівень якості обраного варіанту є відмінним.

Це дозволяє стверджувати, що перевага варіанту № 1 досить переконлива.

5 ОЗНАЙОМЛЕННЯ З ВИБРАНОЮ ТЕХНОЛОГІЄЮ

Історія фреймворка почалася з грудня 2011 під ім'ям Skybreak, в січні 2012 проект змінив своє ім'я. У квітні того ж року ліцензія була змінена з GNU GPL на MIT. У червні Джефф Шмідт (Geoff Schmidt), CEO Meteor Developer Group оголосив про отримання фінансування в розмірі 11.2 млн доларів, в основному від венчурного фонду Andreessen Horowitz. Завдяки цьому Рід Джонсон, творець Spring Framework, який перед тим залишив SpringSource і VMWare, зміг приступити до розробки Meteor, як основній роботі.

Meteor – це платформа для створення так званих real-time web apps – сучасних веб-додатків. По суті, Meteor – це шар між інтерфейсом вашим додатком і його базою даних, який стежить за їх синхронізацією.

Оскільки фреймворк побудований на основі Node.js, то JavaScript використовується як на клієнті, так і на сервері. І більш того, Meteor дозволяє використовувати один і той же код і на клієнті, і на сервері.

В результаті всього цього ми отримуємо дуже потужну, і при цьому просту у використанні платформу, так як більшість стандартних рутин і труднощів створення веб-додатків вже реалізовані з коробки.

Meteor, нова платформа для створення web-додатків, отримує все більш широке міжнародне визнання. Виходячи за рамки простого середовища програмування на JavaScript, Meteor пропонує інноваційні методи конструювання масштабованих, функціонально насичених, інтерактивних web-додатків. Застосування Meteor обіцяє колосальний приріст швидкості розробки за рахунок спрощення моделі програмування і скорочення обсягу коду, який потрібно написати розробнику. Платформа Meteor дозволяє досвідченим архітекторам і розробникам web-додатків скоротити час переходу від концепції до повного впровадження з декількох місяців до лічених днів або тижнів [11].

5.1 Назад у майбутнє

Підхід, пропонований платформою Meteor, в деякому сенсі революційний, але в той же час містить і еволюційні аспекти. Він слід тим же шляхом,
що і одна з найуспішніших в історії категорій програмного забезпечення – електронні таблиці. Типовий приклад електронної таблиці показаний на рисунку 5.1 – таблиця продажів по регіонах з круговою діаграмою.

Якщо змінити в цій таблиці значення продажів по регіонах, то загальна сума продажів (не відображено на малюнку) і відображає відносні частки кругова діаграма миттєво зміняться.

Сьогодні в цьому немає нічого принципово нового чи цікавого. Але уявіть, що за вікном 1983 рік, коли пакет Lotus вперше запропонував ці функції першим користувачам ПК, і ви, можливо, зрозумієте, яке враження вони справили в той час. Ніхто і ніколи раніше не міг домогтися такого ефекту настільки малими зусиллями. І хоча перші електронні таблиці були інтуїтивно зрозумілими, більшість користувачів могли освоїти їх за лічені дні. Електронні таблиці та в наш час є однією з головних рушійних сил продажів комп'ютерів у всьому світі [11].

Рисунок 5.1. Таблиця продажів по регіонах

Перенесемося на три десятиліття вперед. Через тридцять років після появи перших електронних таблиць ви можете спостерігати, як еволюціонувала метафора електронних таблиць з появою платформи Meteor. На рисунку 5.2 показано Web-додаток Sales Portal (портал продажів), створене за допомогою Meteor в 2013 році.

Sales Portal показує поточні показники продажів по регіонах і призводить відповідну кругову діаграму. Якби ви були генеральним директором, ви

могли б контролювати ці показники, а ваші регіональні відділи продажів час від часу могли б їх оновлювати.

Якщо запустити кілька екземплярів браузера і відкрити в них Sales Portal, то всі вони оновляться з урахуванням останніх внесених змін. Значення можна міняти з будь-якого примірника браузера.

Рисунок 5.2. Web-додаток Sales Portal

На рисунку 5.3 показано, як група продажів регіону US Central вибирає і змінює свої показники продажів.

Замість ручного поновлення показників можна створити базу даних зі згрупованими в підмножини показниками продажів, які автономно накопичуються і агрегуються перед оновленням. Візуальне подання додатка Sales Portal ідентично поданням в традиційних електронних таблицях, але на відміну від них Sales Portal:

- забезпечує глобальний доступ в Інтернет через повсюдно поширені браузери;
- надає одночасний доступ кількох користувачів;
- підтримує автоматичну агрегацію і об'єднання даних в базі даних.

Рисунок 5.3. Оновлення показників продажів регіону US Central

На рисунку 5.4 показані оновлений показник продажів регіону US Central і нова кругова діаграма. Будь-який користувач, що працює в Sales Portal одночасно з вами, миттєво побачить ці зміни.

Рисунок 5.4. Кругова діаграма миттєво оновлюється, відображаючи нові показники продажів регіону US Central

Якби ви проектували подібну систему стандартними засобами (наприклад, за допомогою інструментів Java ™), вам потрібні були б значні зусилля з проектування, написання коду і розгортання. Платформа Meteor дозволяє істотно скоротити ці зусилля.

5.2 Реактивне мислення

Однією з основних особливостей електронних таблиць є їх реактивна природа. У наведеному вище прикладі продажів по регіонах при оновленні показника регіональних продажів всі інші компоненти, які залежать від цього значення, перераховуються «на льоту». Якщо залежний компонент відповідає за побудову графічного зображення, наприклад, кругової діаграми, то ця діаграма миттєво перебудовується з новими розмірами секторів. При цьому немає необхідності в написанні обробного залежності коду (який може бути досить складним) або коду для оновлення компонентів, таких як кругова діаграма. Вам потрібно лише оголосити реактивні елементи (показники продажів) і їх залежності (в даному випадку суму продажів і кругову діаграму). Про інше подбає електронна таблиця [12].

А тепер уявіть, що це потрібно зробити за допомогою Web-додатки, і ви отримаєте гарне уявлення про те, як Meteor пропонує спростити створення систем з Web-інтерфейсом.

Проектуючи додаток на платформі Meteor, ви повинні вибрати реактивні елементи, наприклад показники продажів по регіонах. Потім ви компонуєте рівень представлення за допомогою стандартного коду HTML, CSS, бібліотек і компонентів JavaScript на стороні клієнта (таких як JQuery, JQuery UI або Underscore), а також технології створення шаблонів, такий як Хандлёварс. Платформа Meteor відстежує всі залежності реактивних елементів, будує візуальні елементи і перераховує залежності відповідно до останніх оновленими значеннями.

Такий підхід суттєво скорочує обсяг інфраструктурного коду, який потрібно написати, налагодити і протестувати. Вам не доведеться створювати спеціальні Web-сервіси, що синхронізують запити на оновлення, писати код для оновлення бази даних або масивів даних, код, що розсилають повідомлення про зміни іншим підключеним клієнтам, або код для вибірки оновлених значень з бази даних при отриманні повідомлення.

5.3 Переваги та недоліки Метеор

Крім реактивності, що вже само по собі є величезною перевагою, я хотів відзначити ще кілька моментів.

Швидкість розробки. Розробляти під Meteor дійсно приємно. Livereload з коробки, вбудована система білдінга статики. Зручність додавання препроцесорів. Підключення нових файлів на льоту. Деплой вбудованої командою на сервера * .meteor.com для тестування. Тут все зроблено, щоб ви витрачали менше часу на розробку.

Spacebars – переписаний handlebars. Чимось нагадує Reac.js і htmlbars, але працює повільніше, а також володіє магічною здатністю не змінювати html, який ви поміняли за допомогою jquery (наприклад), змінюючи все навколо нього. При цьому не треба обертати в шаблоні цей шматок коду ні в які хелпери. Загалом, якщо ви хочете змінити частину сторінки так, щоб state програми не змінювався, spacebars це може.

Гомогенність. Один з перших фреймворків, який намагається використовувати той факт, що і на сервері, і на клієнті використовується одина і та ж мова, на повну. За замовчуванням, весь ваш код виповнюється і там, і там, а це значить, що вам не потрібно дублювати код, якщо однакова логіка зустрічається на сервері і клієнті [12].

Незважаючи на всі переваги, є і недоліки.

Нестабільність. Часте обновлення версії веб-платформи приводить до частих змін в API, також пристуні баги і регресії, які згодом виправляються.

Чуйність (responsiveness) реалізована трохи викривлено.

Частина логіки поновлення сторінки при появі нових даних реалізується за допомогою винятків (exceptions), і це тягне неприємні наслідки: іноді ви просто не бачите помилку, тому що вона була спіймана якимось внутрішнім оброблювачем Meteor, і тут доводиться займатися налагодженням наосліп.

Також іноді видається неповний stacktrace помилки через наявність асинхронних операцій, але цю проблему можна вирішити за допомогою zones.js.

Невизначеність. Ви постійно перебуваєте в стані незнання того, які дані у вас зараз є, а які ще не прийшли. Тому доведеться ставити купу if. Щоб цього уникнути, намагайтеся не робити дуже складних моделей документів: 1-2 рівня вкладеності. Ну і звичайно, доведеться обробляти всі випадки відсутності даних: ставити спіннери і так далі, але це скоріше добре [12].

6 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ СИСТЕМИ

Meteor – це нова JavaScript-платформа, призначена для автоматизації та спрощення розробки Web-додатків, що діють в режимі реального часу. Вона управляє зв'язком реального часу, використовуючи протокол Distributed Data Protocol (DDP), який підтримується сучасними браузерами за допомогою WebSockets, а браузерами більш ранніх версій – за допомогою механізму long polling Asynchronous JavaScript та XML (Ajax). В обох випадках зв'язок між браузером і сервером залишається прозорою.

Протокол DDP призначений для роботи з колекціями документів JavaScript Serialized Object Notation (JSON), що дозволяє легко створювати, оновлювати, видаляти, запитувати і, звичайно, переглядати документи JSON. Так як DDP – це протокол з відкритим вихідним кодом, він повинен працювати з будь-яким клієнтом або сховищем даних. За замовчуванням він працює з MongoDB.

Фактично, Meteor забезпечує дві бази даних MongoDB: буферну базу даних з боку клієнта і базу даних MongoDB з боку сервера. Коли користувач вносить зміни в дані – наприклад, натиснувши кнопку зберегти – код JavaScript, що виконується в браузері, оновлює відповідний запис в локальній базі даних MongoDB, а потім робить запит DDP до сервера. Код обробляється негайно, як ніби операція виконана успішно, тому що відповіді сервера чекати не потрібно. Тим часом дані на сервері оновлюються у фоновому режимі. Якщо операція на сервері не вдалася, або повертається несподіваний результат, то код JavaScript на стороні клієнта негайно коригує дані відповідно до останнього відповіддю сервера. Це коректування називається компенсацією затримки і забезпечує додаткове відчуття швидкодії у користувача.

Навіть система шаблонів Meteor явно націлена на спрощення зв'язку в режимі реального часу. На більшості web-платформ в код можна легко впроваджувати мову гіпертекстової розмітки (HTML) – або розмітки, еквівалентній HTML, такий як HTML Abstraction Markup Language (Haml). Це дозволяє легко вставляти в сторінки, що відправляються користувачеві, динамічні значення з бази даних. Після цього система повинна стежити за змінами в даних і оновлювати розмітку. Однак система шаблонів в Meteor реєструє, до яких саме даними зверталися через шаблон, і автоматично виконує зворотні ви-

клики, змінюючи цей HTML-код при зміні відповідних даних, що робить шаблони реального часу простими і швидкими [13].

6.1 Архітектура сучасних web-додатків

Meteor призначений для web-додатків з особливою архітектурою, яка показана на рисунку 6.1.

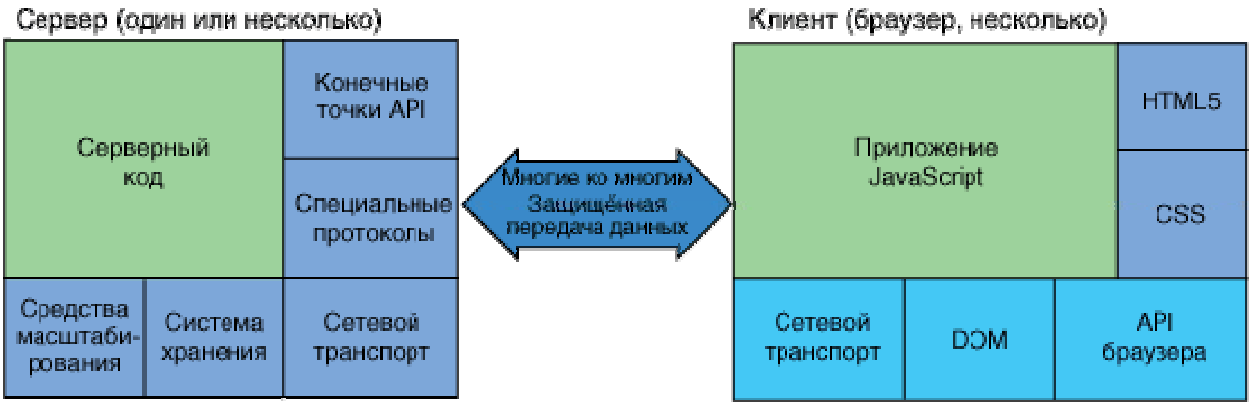


Рисунок 6.1. Архітектура розширених клієнтських інтерактивних web-додатків

Як правило, додатки такого типу складаються з односторінкових інтерфейсів користувача з високим рівнем інтерактивності. Завантаження нових сторінок зазвичай не виконується; замість цього у відповідь на дію користувача миттєво оновлюється частина відображається сторінки – з невеликою мережевий затримкою або взагалі без неї. Односторінковий інтерфейс ні в якій мірі не обмежує додаток, оскільки частини сторінки можуть оновлюватися необмеженим числом способів. Це нагадує поведінку автономних настільних додатків, таких як текстові процесори або електронні таблиці.

Додатки такого типу зазвичай завантажують код JavaScript в браузер клієнта. Цей код управляє взаємодією з користувачем, динамічно маніпулюючи об'єктною моделлю документа (DOM) браузера, змінюючи стиль CSS, генеруючи нові елементи / коди / стилі HTML і використовуючи інші пропоновані браузером API. Всі взаємодія з користувачем контролюється кодом, що працює на стороні клієнта; не завантажується ніяких додаткових кодів HTML або стилів, крім початкового завантаження додатка. Цей же код здійснює взаємний обмін даними між клієнтом і сервером (або серверами) для реалізації функцій програми. По суті, браузер

завантажує і виконує розширений клієнтський додаток (іноді називається товстим клієнтом), написане на JavaScript [14].

На стороні сервера організуються кінцеві точки, які передають і синхронізують дані клієнта в захищеному режимі. Старі сервери можуть використовувати RPC, Web-сервіси на основі XML, служби RESTful або інші виклики RPC в стилі JSON. Сучасні сервери, швидше за все, будуть використовувати спеціальні протоколи, призначені для ефективної передачі даних, стійкі до обривів зв'язку, що підтримують різні види сучасного транспорту і топологічний масштабування.

На рисунку 6.2 показані основні компоненти платформи Meteor версії 0.6.3.1.

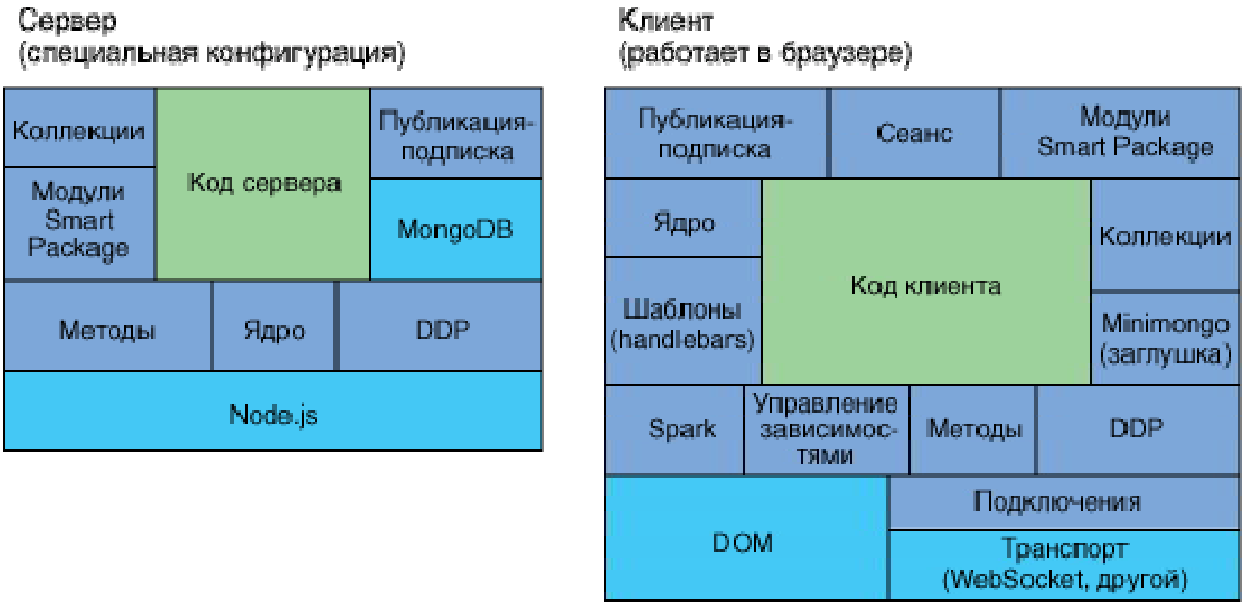


Рисунок 6.2. Внутрішні компоненти платформи Meteor

Протокол DDP підтримує двонаправлений обмін даними між екземплярами клієнта і сервера, а Minimongo є «інтелектуальну» заглушку з локальним кешем даних на стороні клієнта, яка надає знайомий API запитів MongoDB для коду, що працює на стороні клієнта. Spark – це живий механізм обробки HTML, який працює спільно з шаблонами (Handlebars) і системою управління залежностями і призначений для реалізації реактивної відтворення в платформі Meteor [13].

6.2 Архітектура додатків, що працюють з даними великого обсягу

Односторінкового динамічна парадигма платформи Meteor з високим рівнем інтерактивності фактично налаштована на рішення певного класу задач. Одним з аспектів візуалізації великих обсягів даних є потреба в інтерактивних інформаційних панелях, які оновлюються в міру надходження результатів. Крім того, подібна інформаційна панель може застосовуватися для створення черги завдань MapReduce і відстеження їх виконання.

Реактивні додатки Meteor дозволяють створювати інтерактивні інтерфейси для роботи з терабайтами і петабайтами даних – аналогічно електронної таблиці, яка може запропонувати інтерактивне оновлення, підсумовування, деталізацію і спеціальні представлення відносно невеликих обсягів даних. Архітектура такої системи показана на рисунку 6.3.

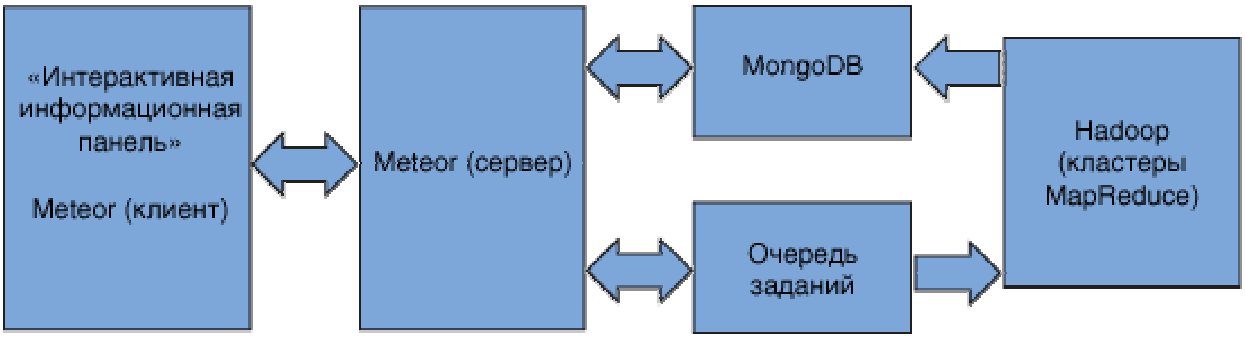


Рисунок 6.3. Архітектура інформаційної панелі Meteor для даних великого обсягу

На рисунку 6.3 код Meteor на стороні клієнта пропонує уявлення сховища великого обсягу даних у вигляді реактивної інформаційної панелі. Вирішення генерують спеціальні завдання MapReduce, які ставляться в чергу (можливо, за допомогою методів Meteor) на стороні сервера для виконання в кластерах Hadoop. По завершенні виконання завдань результати об'єднуються в екземплярі MongoDB через Meteor. Компонент публікації-підписки сервера Meteor виявляє зміну даних і передає оновлені дані клієнтам які підписалися [14].

6.3 Мережева архітектура Метеор-додатка

Додаток Метеор, з точки зору браузерів, проксі-серверів, маршрутизаторів і інших мережевих компонентів, є, по суті, звичайним веб-додатком. Хоча, насправді, Метеор-додаток складається з двох головних частин: части-

на, яка працює всередині браузера і частина, яка працює як сервер (рис. 6.4) . Ці дві частини налаштовані таким чином, щоб взаємодіяти один з одним способом, характерним для багатьох сучасних веб-додатків (таких, як Gmail або Trello).

Метеор дозволяє розробникам створювати додатки не піклуючись про складнощі клієнт-серверної взаємодії. Якщо копнути глибше, Метеор обробляє три типи запитів. Ось вони:

- статичні файли;
- повідомлення по протоколу DDP;
- HTTP запити.

Статичні файли – це картинки та інші подібні ресурси з папки / public. Метеор обробляє ці файли автоматично при запуску.

Додатково, Метеор мініфіцірує і склеює весь JavaScript (включаючи шаблони, які попередньо компілюються в JavaScript) і CSS файли, віддаючи їх як статичні.

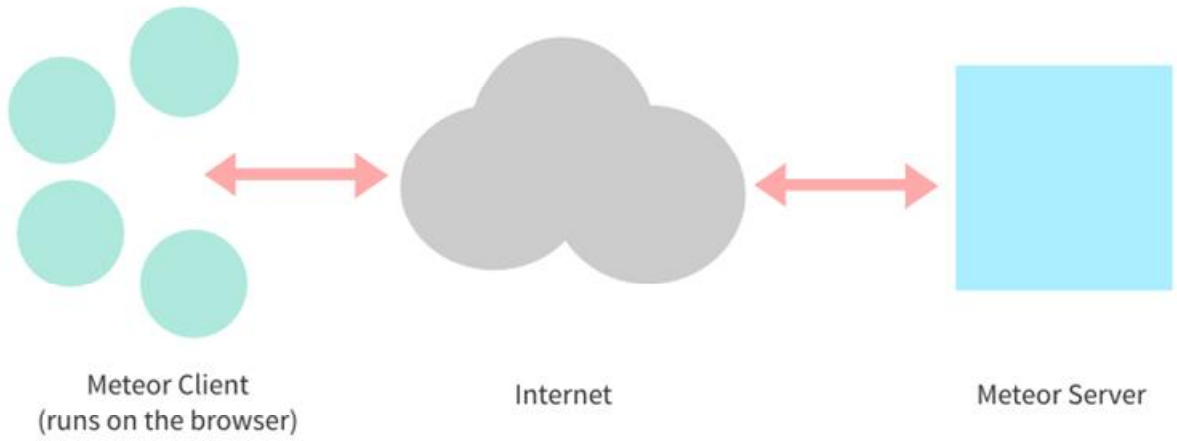


Рисунок 6.4. Мережева архітектура Метеор-додатка

DDP повідомлення. DDP – це протокол, який Метеор використовує для взаємодії клієнтської і серверної частини. Всі клієнтські підписки на дані, віддалений виклик процедур і операції MongoDB – все це відбувається з використанням протоколу DDP. При цьому – це досить легкий протокол. Повідомлення можна переглядати за допомогою зручного інструменту – ddp-analyzer [14].

HTTP запити. Метеор може обробляти HTTP-запити, подібно до інших звичайних додатків. Наприклад, завантаження файлів обробляється Метеором, як HTTP-запити.

Всередині у Метеора два сервера. Хоча Метеора прослуховує тільки один порт, всередині він працює, як два окремих сервера (рис. 6.5):

- HTTP сервер;
- DDP сервер.

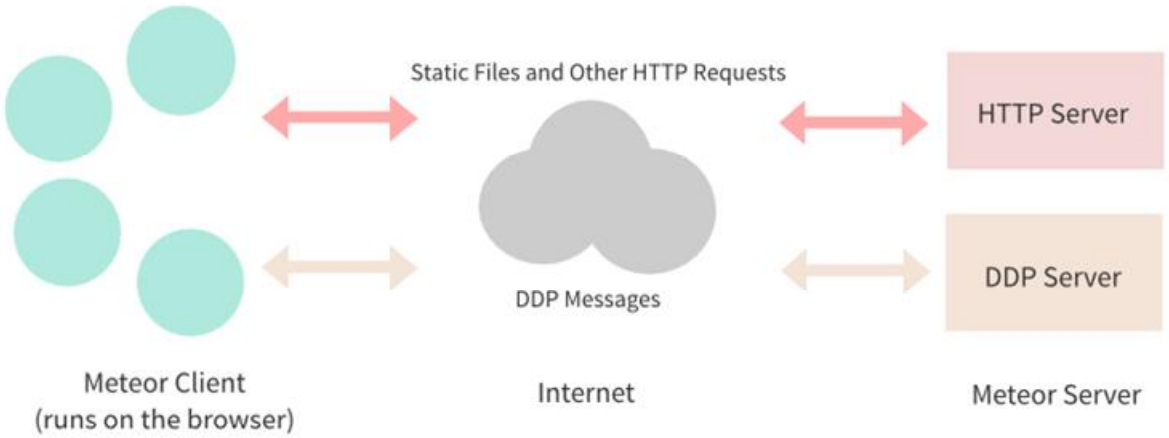


Рисунок 6.5. Метеор має всередині два сервера

HTTP сервер використовується для передачі статичних файлів і обробки HTTP запитів. Для цих цілей в Метеор використовується connect Node.js модуль.

DDP сервер обробляє всі публікації даних, MongoDB операції і метеор методи. Метеор використовує SockJS, як транспорт. По суті, DDP – це SockJS сервер.

6.4 MongoDB

MongoDB (від англ. Humongous – величезний) – документоорієнтованих система управління базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць. Класифікована як NoSQL, використовує JSON-подібні документи і схему бази даних. Написана на мові C++. MongoDB – це система управління базами

даних, яка значно відрізняється від MySQL. Основна різниця полягає в тому, що в MySQL запити пишуться на мові SQL, а в MongoDB на BSON (бінарний JSON). Це означає, що робота з цією системою може здійснюватися в основному через JavaScript вираження.
Також MongoDB включає в себе власну утиліту для виконання команд, спрямованих на роботу з БД.

MongoDB реалізує новий підхід до побудови баз даних, де немає таблиць, схем, запитів SQL, зовнішніх ключів і багатьох інших речей, які притаманні об'єктно-реляційним базам даних.

З часів динозаврів було звичайною справою зберігати всі дані в реляційних базах даних (MS SQL, MySQL, Oracle, PostgresSQL). При цьому було не так важливо, а чи підходять реляційні бази даних для зберігання даного типу даних чи ні.

На відміну від реляційних баз даних MongoDB пропонує документоорієнтовану модель даних, завдяки чому MongoDB працює швидше, має кращу масштабованість, її легше використовувати.

Але, навіть враховуючи всі недоліки традиційних баз даних і гідності MongoDB, важливо розуміти, що завдання бувають різні і методи їх вирішення бувають різні. В якійсь ситуації MongoDB дійсно поліпшить продуктивність вашої програми, наприклад, якщо треба зберігати складні за структурою дані. В іншій же ситуації краще буде використовувати традиційні реляційні бази даних. Крім того, можна використовувати змішані підхід: зберігати один тип даних в MongoDB, а інший тип даних – в традиційних БД.

6.4.1 Формат даних в MongoDB

Вся система MongoDB може представляти не тільки одну базу даних, що знаходиться на одному фізичному сервері. Функціональність MongoDB дозволяє розташувати кілька баз даних на декількох фізичних серверах, і ці бази даних зможуть легко обмінюватися даними і зберігати цілісність.

Одним з популярних стандартів обміну даними та їх зберігання є JSON (JavaScript Object Notation). JSON ефективно описує складні за структурою дані. Спосіб зберігання даних в MongoDB в цьому плані схожий на JSON, хоча формально JSON не використовується. Для зберігання в MongoDB застосовується формат, який називається BSON (Бісон) або скорочення від binary JSON. BSON дозволяє працювати з даними швидше: швидше виконується пошук і обробка. Хоча треба зазначити, що BSON на відміну від

зберігання даних в форматі JSON має невеликий недолік: в цілому дані в JSON-форматі займають менше місця, ніж в форматі BSON, з іншого боку, даний недолік з лишком окупається швидкістю [13].

6.4.2 Кросплатформеність

MongoDB написана на C ++, тому її легко перенести на найрізноманітніші платформи. MongoDB може бути розгорнута на платформах Windows, Linux, MacOS, Solaris. Можна також завантажити вихідний код і самому скомпілювати MongoDB, але рекомендується використовувати бібліотеки з офіційного сайту.

6.4.3 Документи замість рядків

Якщо реляційні бази даних зберігають рядки, то MongoDB зберігає документи. На відміну від рядків документи можуть зберігати складну за структурою інформацію. Документ можна уявити як сховище ключів і значень. Ключ являє просту мітку, з яким асоційоване певний шматок даних.

Однак при всіх відмінностях є одна особливість, яка зближує MongoDB і реляційні бази даних. У реляційних СУБД зустрічається таке поняття як первинний ключ. Це поняття описує якийсь стовпець, який має унікальні значення. У MongoDB для кожного документа є унікальний ідентифікатор, який називається _id. І якщо явно не вказати його значення, то MongoDB автоматично згенерує для нього значення.

Кожному ключу зіставляється певне значення. Але тут також треба враховувати одну особливість: якщо в реляційних базах є чітко окреслена структура, де є поля, і якщо якесь поле не має значення, йому (в залежності від налаштувань конкретної бд) можна привласнити значення NULL. У MongoDB все інакше. Якщо якомусь ключ не порівнювати значення, то цей ключ просто опускається в документі і не вживається.

Колекції. Якщо в традиційному світі SQL є таблиці, то в світі MongoDB є колекції. І якщо в реляційних БД таблиці зберігають однотипні жорстко структуровані об'єкти, то в колекції можуть містити найрізноманітніші об'єкти, що мають різну структуру і різний набір властивостей [12].

6.5 Проектування проекту

Перший етап процесу проектування бази даних називається концептуальним проектуванням бази даних. Він полягає у створенні концептуальної моделі даних для аналізованої частини системи. Ця модель даних створюється на основі функціональних вимог користувачів. Концептуальне проектування бази даних абсолютно не залежить від таких подробиць її реалізації, як тип обраної СУБД, набір створюваних прикладних програм, що використовуються мови програмування, тип обраної обчислювальної платформи, а також від будь-яких інших особливостей фізичної реалізації. Концептуальне проектування створення концептуального уявлення бази даних, що включає визначення типів найважливіших сутностей та існуючих між ними зв'язків і атрибутів [16].

Запишемо послідовність етапів проектування концептуальної моделі даних:

- визначення сутностей;
- визначення взаємозв'язків між сутностями;
- визначення атрибутів сутностей;
- завдання первинних та альтернативних ключів.

Побудуємо концептуальну модель бази даних пекарні. Перейдемо до виконання послідовності проектування. Оберемо спроектовані таблиці бази даних та запишемо їх головні сутності.

Сутність КЛІЄНТИ містить наступні атрибути (табл. 6.1): id_клієнта, ім’я підприємства, ідентифікаційний номер платника податків, адресу, ім’я клієнта, контактний телефон.

Таблиця 6.1 – Сутність «КЛІЄНТИ»

№	Атрибут	Тип
1	_id	Number
2	name	String
3	inn	String
4	address	String
5	contactName	String
6	contactPhone	Number

Сутність ВОДІЇ містить наступні атрибути (табл. 6.2): id_водія, ім’я, телефон.

Таблиця 6.2 – Сутність «ВОДІЇ»

№	Атрибут	Тип
1	_id	Number
2	name	String
3	phone	Number

Сутність КОРИСТУВАЧІ містить наступні атрибути (табл. 6.3): id_користувача, дата створення, email-адреса, роль.

Таблиця 6.3 – Сутність «КОРИСТУВАЧІ»

№	Атрибут	Тип
1	_id	Number
2	createdAt	Date
3	emails	Array
4	roles	Object

Сутність ЖУРНАЛ містить наступні атрибути (табл. 6.4): id_запису, id_користувача зробившого запис, дата, час запису, рівень, опис.

Таблиця 6.4 – Сутність «ЖУРНАЛ»

№	Атрибут	Тип
1	_id	Number
2	userId	Number
3	date	Date
4	timestamp	Number
5	level	String
6	message	String

Сутність РОЛІ містить атрибути (табл. 6.5): id_ролі, назва ролі.

Таблиця 6.5 – Сутність «РОЛІ»

№	Атрибут	Тип
1	_id	Number
2	name	String

Сутність ІНГРЕДІЄНТИ містить наступні атрибути (табл. 6.6): id_інгредієнту, назва інгредієнту, поточний баланс, початковий баланс.

Таблиця 6.6 – Сутність «ІНГРЕДІЄНТИ»

№	Атрибут	Тип
1	_id	Number
2	name	String
3	currentBalance	Number
4	initBalance	Number

Сутність ЗАВДАННЯ ПЕКАРЯМ містить наступні атрибути (табл. 6.7): id_завдання, дата, опис, закази, продукти.

Таблиця 6.7 – Сутність «ЗАВДАННЯ ПЕКАРЯМ»

№	Атрибут	Тип
1	_id	Number
2	date	Date
3	description	String
4	orders	Array
5	products	Object

Сутність ПРОДУКТИ містить наступні атрибути (табл. 6.8): id_продукта, назва продукта, опис, інгредієнти.

Таблиця 6.8 – Сутність «ПРОДУКТИ»

№	Атрибут	Тип
1	_id	Number
2	name	String
3	description	String
4	ingredients	Array

Сутність ЗАЯВКИ містить наступні атрибути (табл. 6.9): id_заявки, дата, дата заявки, клієнти, ім’я клієнта, опис, продукти.

Таблиця 6.9 – Сутність «ЗАЯВКИ»

№	Атрибут	Тип
1	_id	Number
2	date	Date
3	orderDate	Date
4	customerId	Number
5	customerName	String
6	description	String

7	products	Array

Описавши, всі головні сутності й атрибути розроблюваної бази даних для системи управління пекарнею можна визначити зв'язки між сутностями. Уявімо базу даних у вигляді моделі «сутність-зв'язок». Тип використовуваного зв'язку «один-до-багатьох». Діаграма «сутність-зв'язок» представлена на рисунку 6.1.

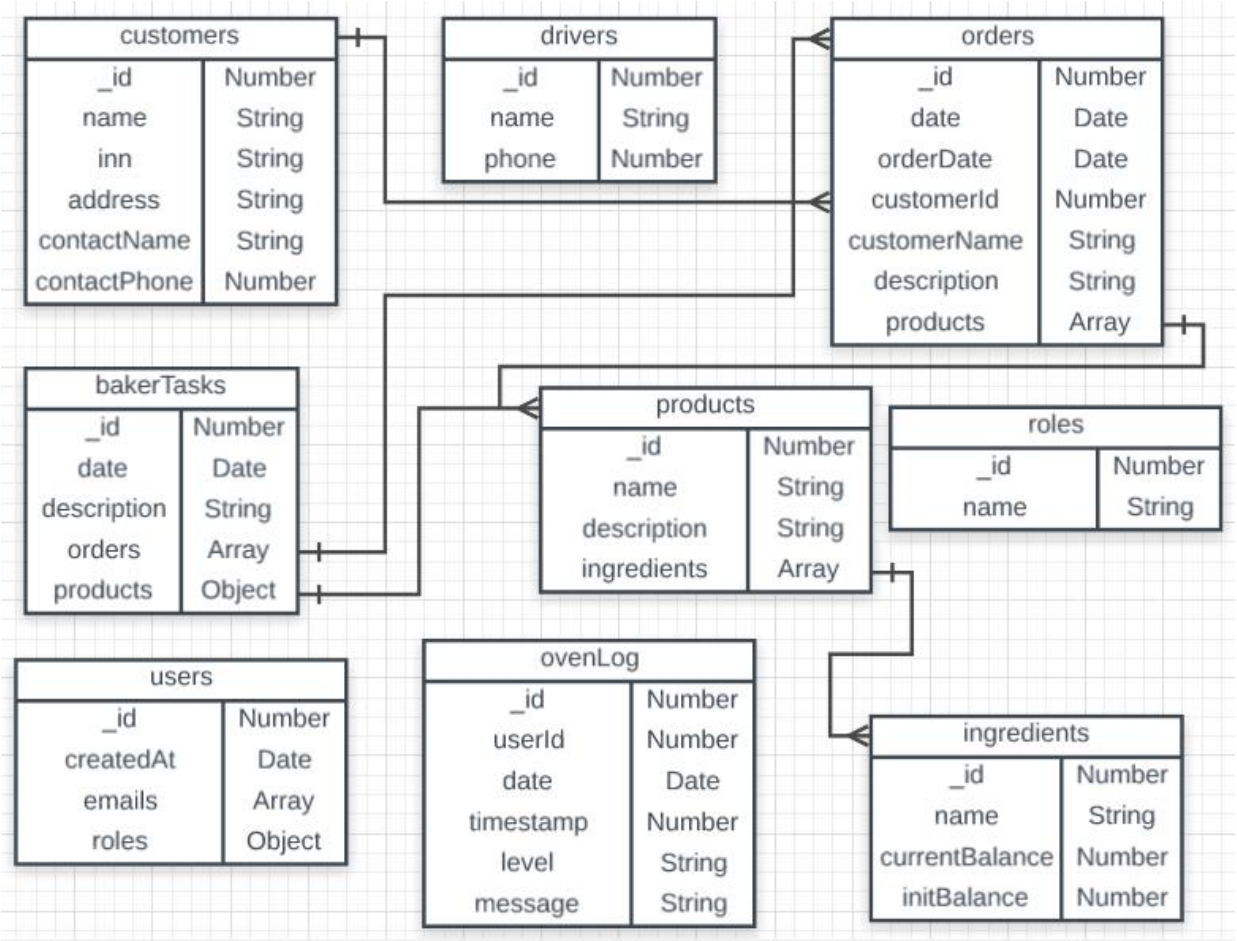


Рисунок 6.1 – Модель «сутність-зв'язок»

Між сутностями «КЛІЄНТИ» та «ЗАЯВКИ» існує зв’язок один-до-багатьох: багато заявко належать до одного клієнта.

Між сутностями «ЗАВДАННЯ ПЕКАРЯМ» та «ЗАЯВКИ» існує зв’язок один-до-багатьох: декілька заявок можуть належати до одного завдання.

Між сутностями «ЗАВДАННЯ ПЕКАРЯМ» та «ПРОДУКТИ» існує зв’язок один-до-багатьох, тобто багато продуктів можуть використовуватися для одного завдання.

Між сутностями «ЗАЯВКИ» та «ПРОДУКТИ» існує зв’язок один-до-багатьох, тобто багато продуктів можуть використовуватися для однієї заявки.

Між сутностями «ПРОДУКТИ» та «ІНГРЕДІЄНТИ» існує зв’язок один-до-багатьох, тобто багато інгредієнтів можуть входити в склад одного продукта.

У результаті проведеного, в ході данної роботи, проектування системи управління пекарнею визначена архітектура системи та розроблена база даних системи.

7 ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ

Результат роботи системи управління пекарнею будемо розглядати на прикладі тестового WEB-додатка.

Система представляє собою робоче місце для керуючого пекарнею, а також для диспетчера.

Роль керівника пекарні буде виконувати адміністративні функції:

- додавати нових користувачів системи;
- редагувати інформацію користувачів системи;
- переглядати всіх зареєстрованих користувачів;
- додавати і редагування водіїв;
- додавати і редагування клієнтів;
- огляд журналу дій користувачів;
- слідкувати за роботою системи;

Диспетчер буде виконувати:

- прийом замовлень на наступний день;
- створення завдання для пекарів;
- додавання і редагування інгредієнтів;
- створення і редагування продуктів;
- складання виробничого завдання для пекарів;
- прийом готової продукції на скла;
- розподілення готової продукції по автомобілям, друк завдання для водіїв;
- прийом повернень;

7.1 Структура системи

Система управління пекарнею є real-time додатком, яка має розподілену структуру, яка складається з вертикального меню, яке знаходить ліворуч від основної частини; шапки системи, яка знаходиться в верхній частині інтерфейсу; основної частини, де виводиться основний контент системи та підвалу, який прікріпленний в нижній частині інтерфейсу. Головна сторінка тестової системи представлена на рис. 7.1.

Вся сторінка залишається статичною, динамічна тільки основна частина, де відбувається підключення шаблонів відповідних сторінок.

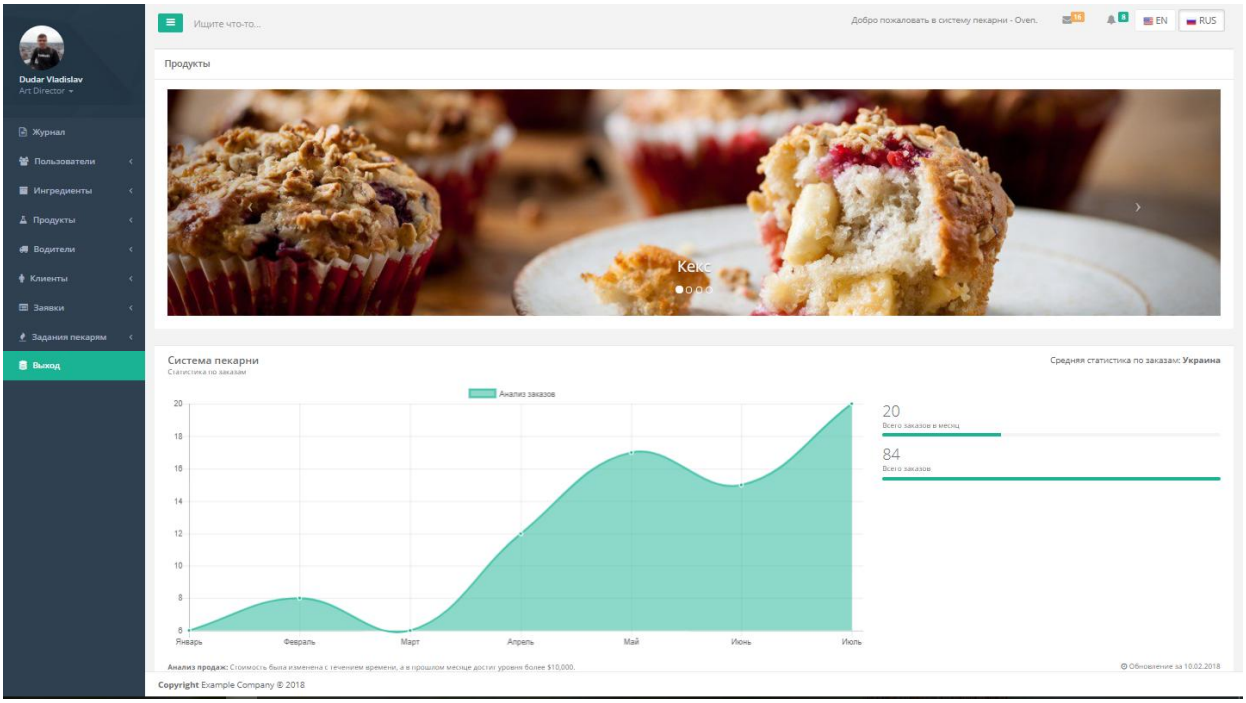
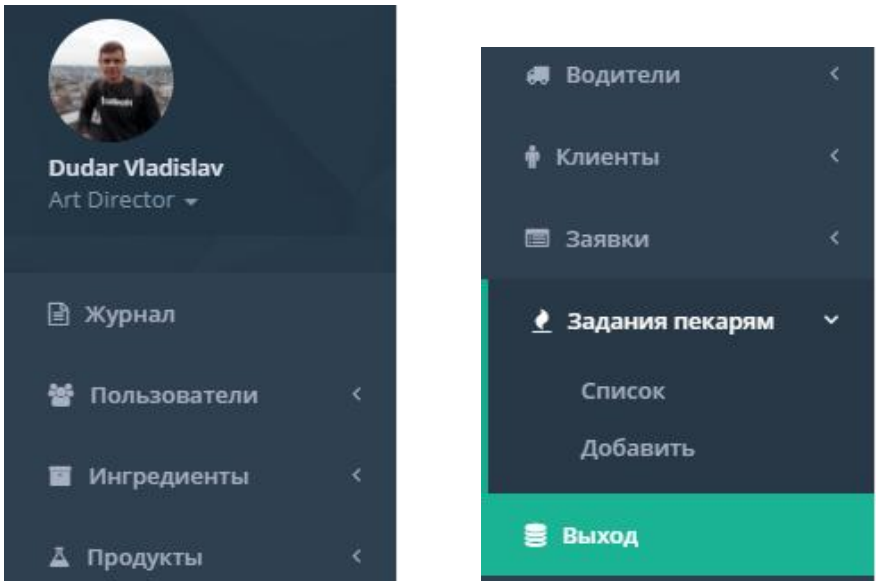


Рисунок 7.1 – Головна сторінка системи управління пекарнею

У вертикальному меню, знаходяться наступні посилання: «Журнал», «Пользователи», «Ингредиенты», «Продукты», «Водители», «Клиенты», «Заявки», «Задания пекарям». Вертикальне меню представлено на рис. 7.2.



а) б)

Рисунок 7.2 – Вертикальне меню:
а – перша частина, б – друга частина

В верхній частині інтерфейсу ситеми знаходиться пошук по системі, привітання, соціальні кнопки, а також посилання на дві версії системи на різних мовах, header інтерфейсу представленна на рис. 7.3.

Інтерфейс система управління пекарнею організован двома мовами – англійською та російською.

Рисунок 7.3 – «Шапка» інтерфейсу

В нижній частині інтерфейсу знаходиться footer, на якому міститься інформація про розробника системи та соціальні посилання (рис. 7.4).

Рисунок 7.4 – «Підвал» інтерфейсу

Основна частина динамічна – це означає що вона буде змінювати свій зміст відносно того яка сторінка відкрита. Приклад головної сторінки представленно на рис. 7.5.

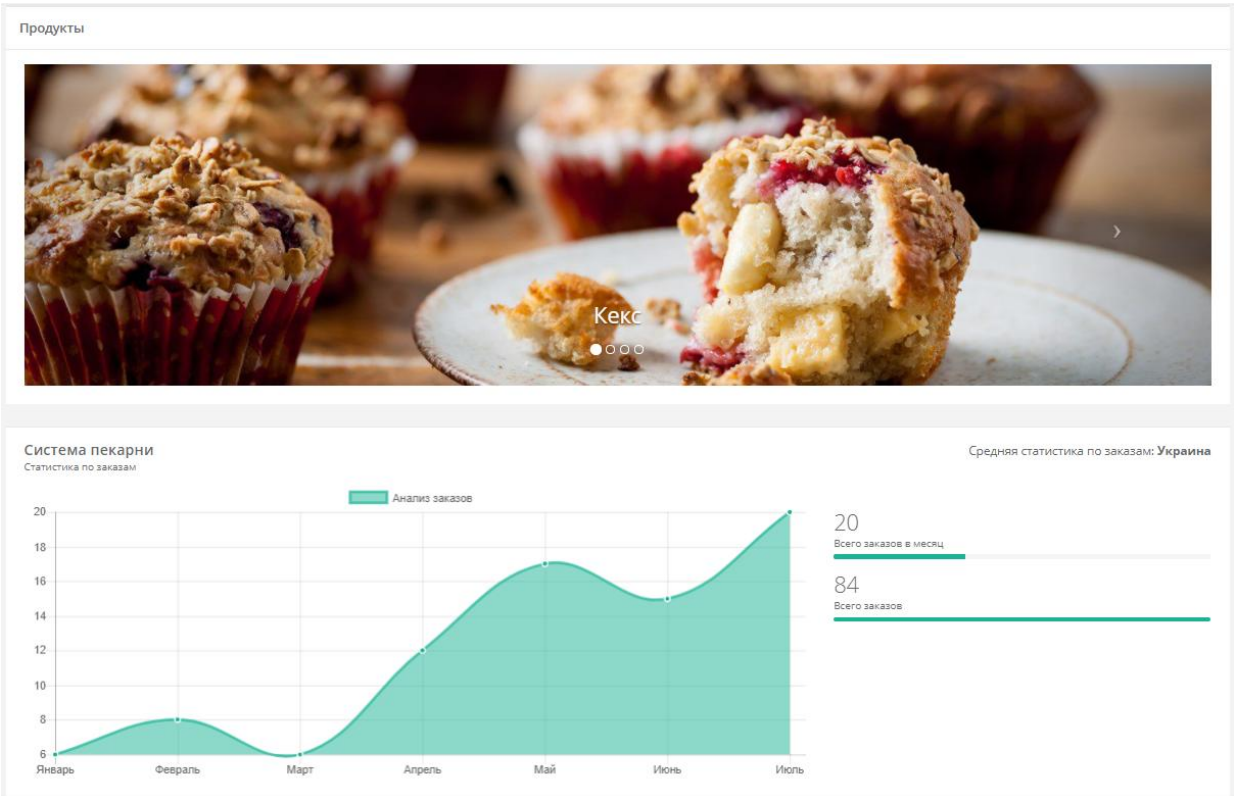


Рисунок 7.5 – Початкова сторінка

В верхній частині знаходиться слайдер, в якому зображені товари виробленні пекарнею, код слайдеру представленний у додатку А.

В нижній частині виводиться графік, який відображає статистику заказів пекарні, по осі X відображен час, по осі Y кількість заказів. Також виводиться інформація скільки було заказів в поточному місяці та зашалом.

7.2 Огляд основних розділів системи

Кожен розділ в системі має хлібні крихти (навігаційна ланцюжок, англ. Breadcrumbs) – це елемент навігації по сайту, який представляє собою шлях від кореня сайту, до поточної сторінки, на якій в даний момент знаходиться користувач (рис. 7.6).

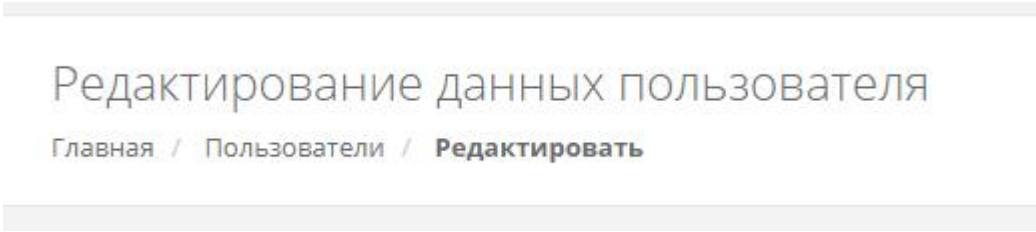


Рисунок 7.6 – Хлібні крихти

Також кожна система для кожної дії добавлення, редагування видалення, підтвердження викликає сповіщення, яке працює завдяки установленного плагіну toastr (рис. 7.7).

Рисунок 7.7 – Сповіщення

7.2.1 Розділ «Журнал»

Сторінка журналу містить в собі записи всіх змін, які проходять в системі, чи то зміна ім’я користувача, чи зміна змісту продукта, також є календар, в якому адміністратор системи може подивитися всі зміни за конкретну дату.

Використовувася календар DatePicker – потужний і простий у використанні jQuery плагін, який дозволяє додавати до простого текстового поля віджет – календар для вибору дати, код реалізації представлений в додатку Б.

Робота з датою і часом найчастіше досить обтяжлива, тому окрема JavaScript бібліотека для маніпуляції датами була б дуже корисною. Є чудова бібліотека Moment.js, що дозволяє затверджувати, парсити і управляти датами і часом. Ця бібліотека спрощує маніпуляцію і валідацію дат і часу, спрощує роботу розробникам додатків.

Для роботи з записами всіх змін було доданно до системи наступні пакети: ostrio:logger та розширення для БД ostrio:loggermongo. Кожна зміна відображена різними кольорами відносно її рівня (рис. 7.8).

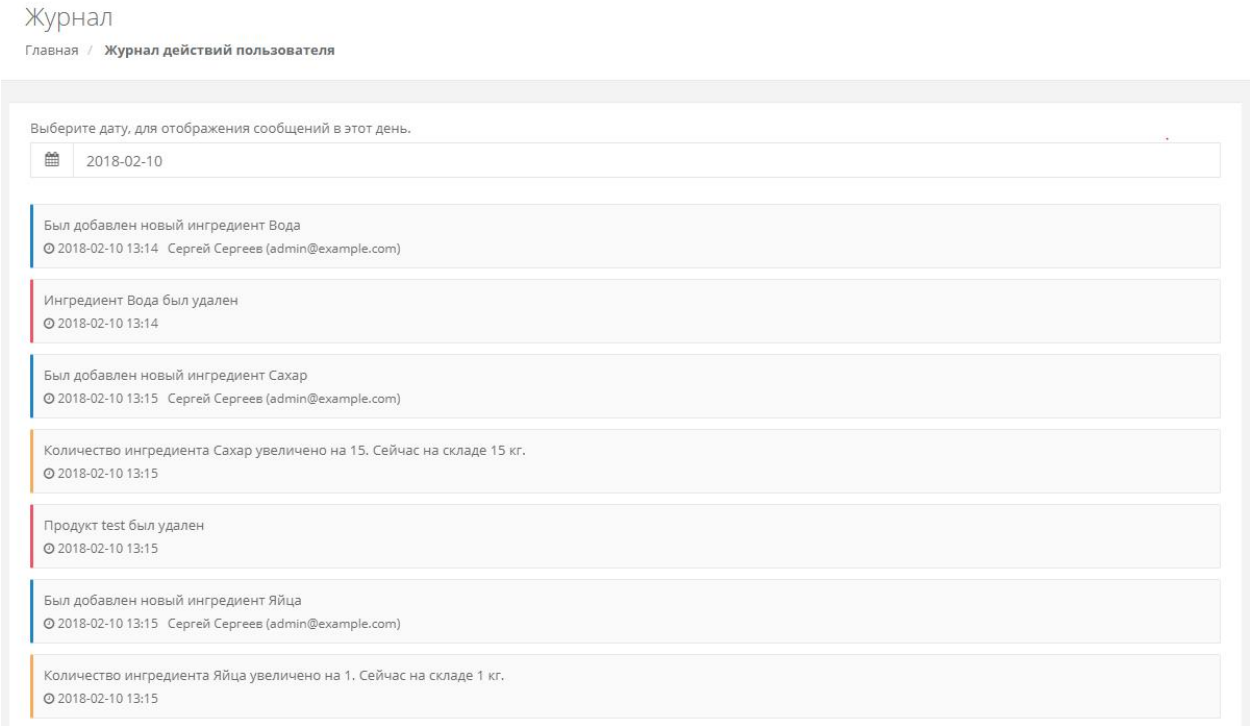


Рисунок 7.8 – Розділ «Журнал»

7.2.2 Розділ «Користувачі»

Данний розділ має два посилання, одне на список існуючих користувачів, а інше посилання на створення нового користувача.

Права на створення та перегляд користувачів має тільки адміністратор системи. Для створення нового користувача необхідно заповнити форму.

Сторінка створення представленна на рис. 7.9.

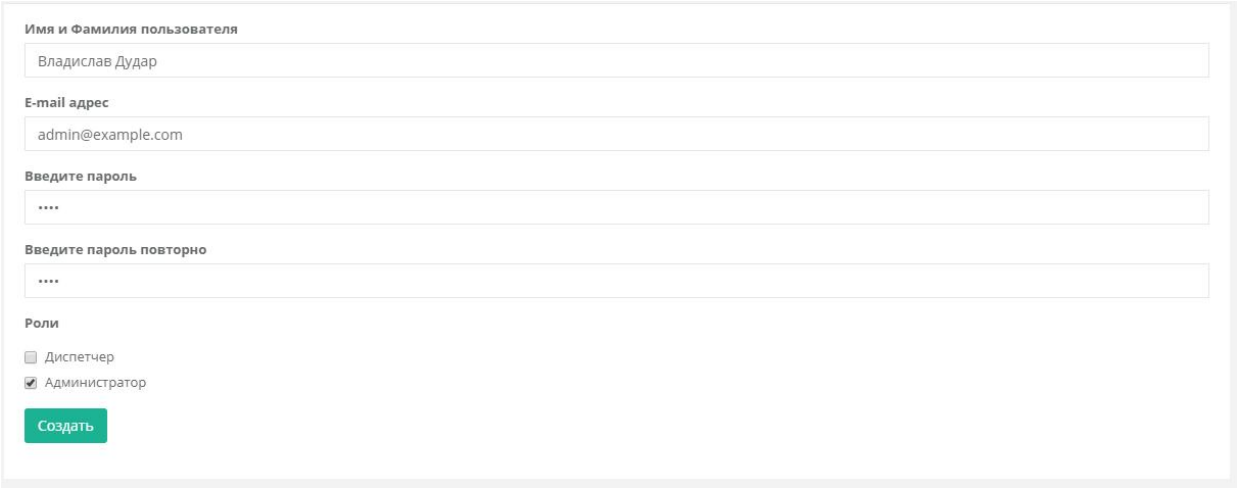


Рисунок 7.9 – Розділ «Створення користувача»

Сторінка перегляду існуючих користувачів зображена на рис. 7.10. На ній виводяться всі користувачі та інформація про них, а саме якою ролю наділен користувач, його електрона адреса, ім’я та фамілія, також є посилання на редагування користувача.

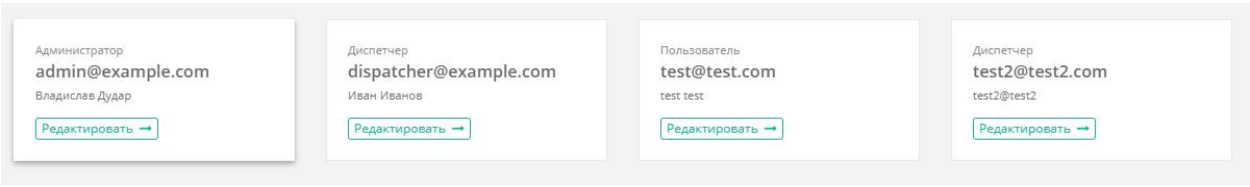


Рисунок 7.10 – Розділ «Перегляд користувачів»

Якщо натиснути на посилання редагування відкриється нова сторінка, в якій буде форма схожа на форму створення нового користувача, але в її полях буде інформація про поточного користувача, яку можно злегкістью редагувати. Після натискання на кнопку збереження, нова інформація замінить стару в базі данних.

Для роботи з кодуванням паролів був установлен пакет accounts-password. Пакет містить повну систему аутентифікації на основі пароля. В додаток до основного процесу входу в систему з ім'ям користувача та паролем, він також підтримує повідомлення по електронній пошті, включаючи перевірку адреси та відновлення пароля.

Сервер метеор зберігає паролі з використанням алгоритму bcrypt. Це допомагає захистити від невдачливих витоків паролів, якщо база даних сервера під атакою.

7.2.3 Розділ «Інгредієнти»

Розділ призначений для створення та перегляду інгредієнтів. Це є створення віртуального інгредієнту, його переносу в облік системи. Наприклад у пекарні є мука із зазначенням ваги, так цей розділ необхідний для переносу в електронний вид інгредієнта для роботи з ним, створення з нього продуктів.

Створення інгредієнта показано на рис. 7.11. Необхідно заповнити поле в формі.

Після створення інгредієнту, система автоматично переадресує адміністратора на сторінку перегляду всіх створенних інгредієнтів. Коженому інгредієнту необхідно задати вагу.

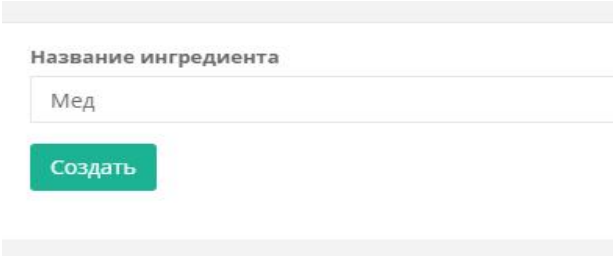


Рисунок 7.11 – Розділ «Створення інгредієнту»

Після того як інгредієнт буде добавлен в індекаторі залишку буде 0%, тобто в базі данних для цього інгредієнту ще не добавленна вага (рис 7.12).

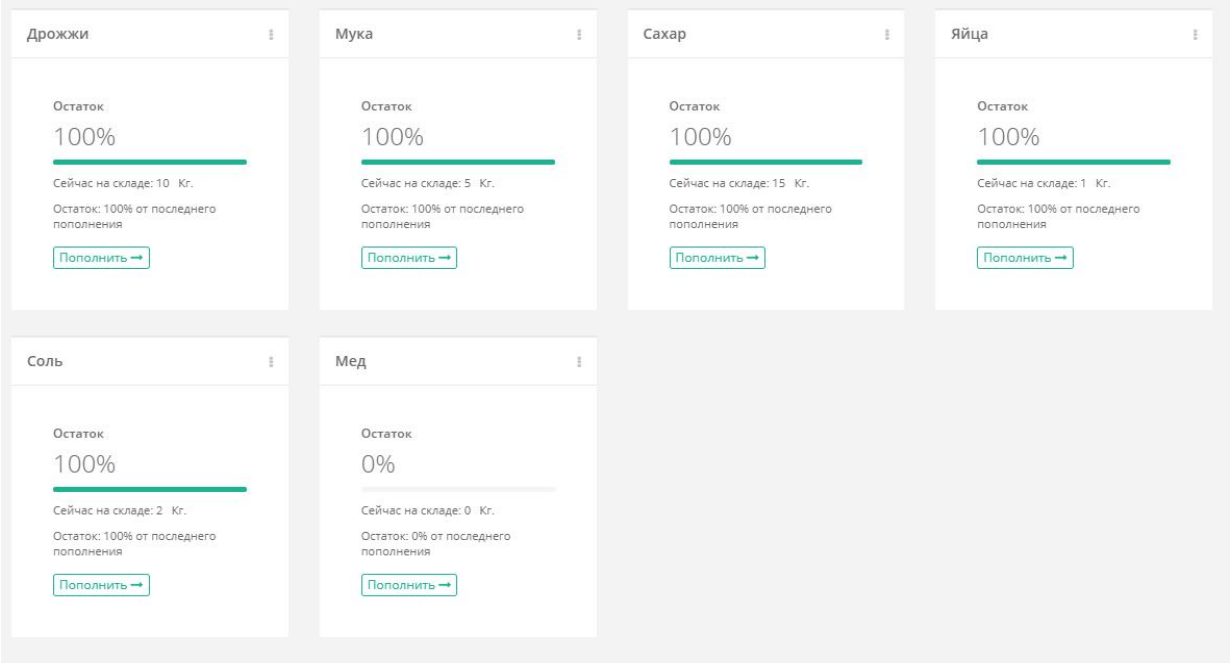


Рисунок 7.12 – Розділ «Список інгредієнтів»

Натиснемо на кнопку «Пополнить» для того щоб задати вагу. Після цього з’явиться модальне вікно, в якому необхідно встановити вагу для поточного інгредієнта. За замовчуванням встановленно 1.0 кг. Далі необхідно зберегти встановленну вагу натиснувши на кнопку «Сохранить», якщо необхідно відмінити поповненя, можно це зробити через кнопку «Отмена» (рис. 7.13).

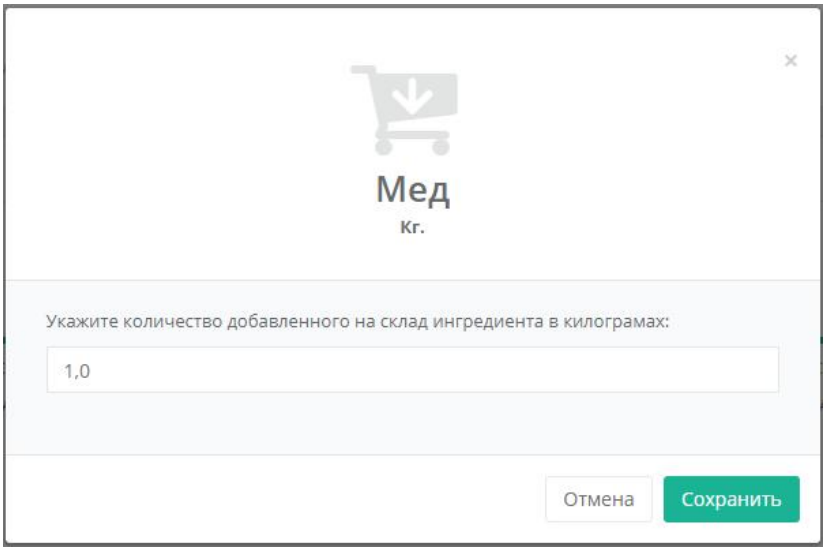


Рисунок 7.13 – Модальне вікно

Одразу після встановлення ваги інгредієнту в полі залишок буде встановленно 100%, тобто величина залишку на складі. Ця величина буде змінюватися, коли інгредієнт буде використовуватися (рис. 7.14).

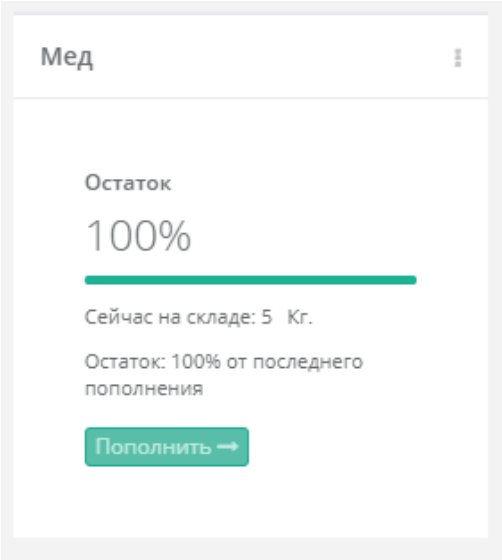


Рисунок 7.14 – Створенний інгредієнт

Системою передбачено видалення інгредієнту. Якщо натиснути по іконці, яка знаходиться в верхній частині інгредієнту, то з’явиться посилання на видалення поточного інгредієнту (рис. 7.15).

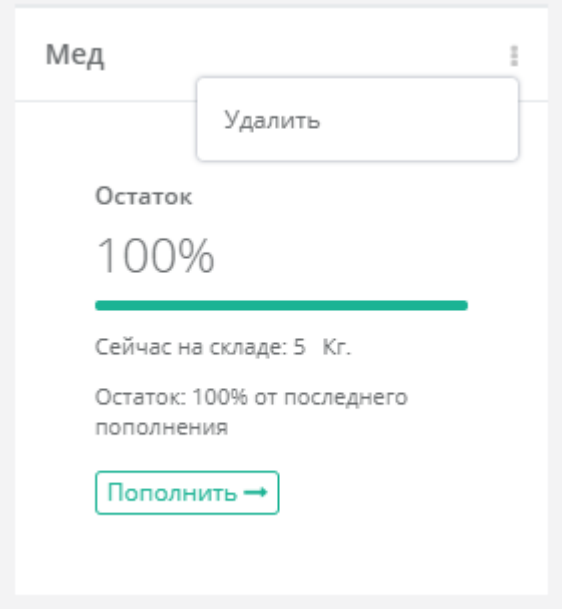


Рисунок 7.15 – Видалення інгредієнта

Одразу після натискання по посиланню «Удалить» відкриється модальне вікно з питання підтвердити видалення (рис. 7.16).

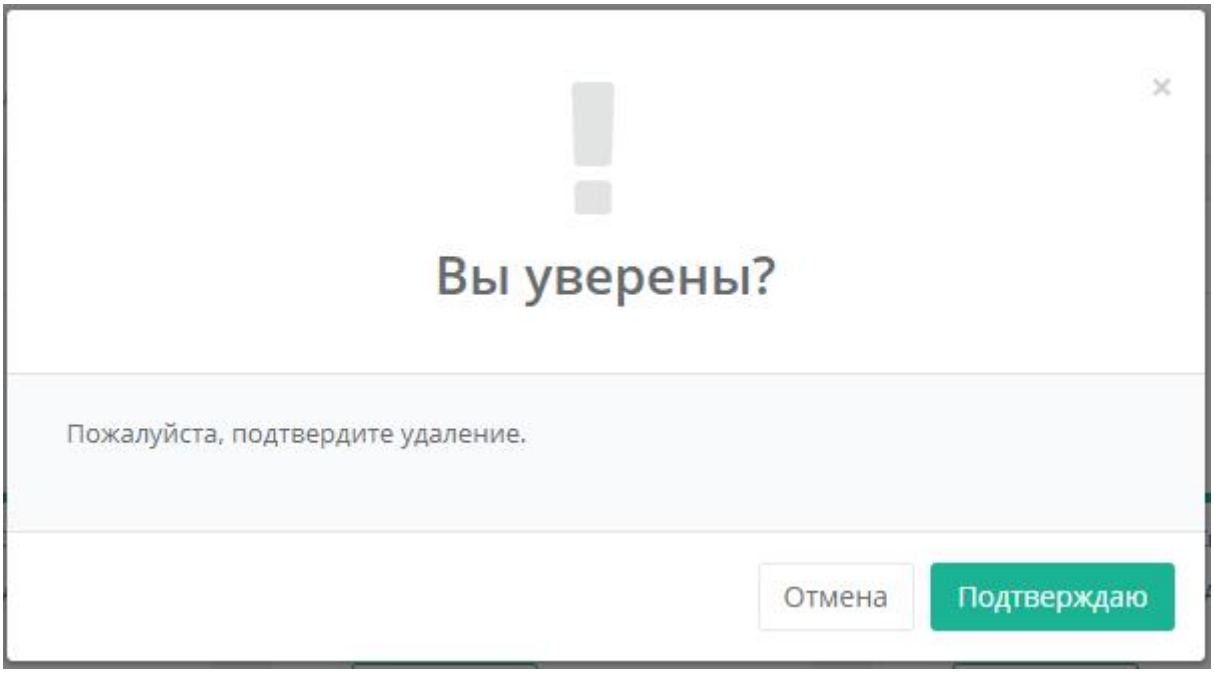


Рисунок 7.16 – Підтвердження

7.2.4 Розділ «Продукти»

Данний розділ також включає два посилання на створення та перегляду продуктів.

Створення продукта показано на рис. 7.17. Необхідно заповнити поле назва продукта, а також вибрати з яких інгредієнтів він буде створений.

Рисунок 7.17 – Створення продукта

Після створення продукта, система автоматично переадресує адміністратора на сторінку перегляду всіх створенних продуктів (рис. 7.18).

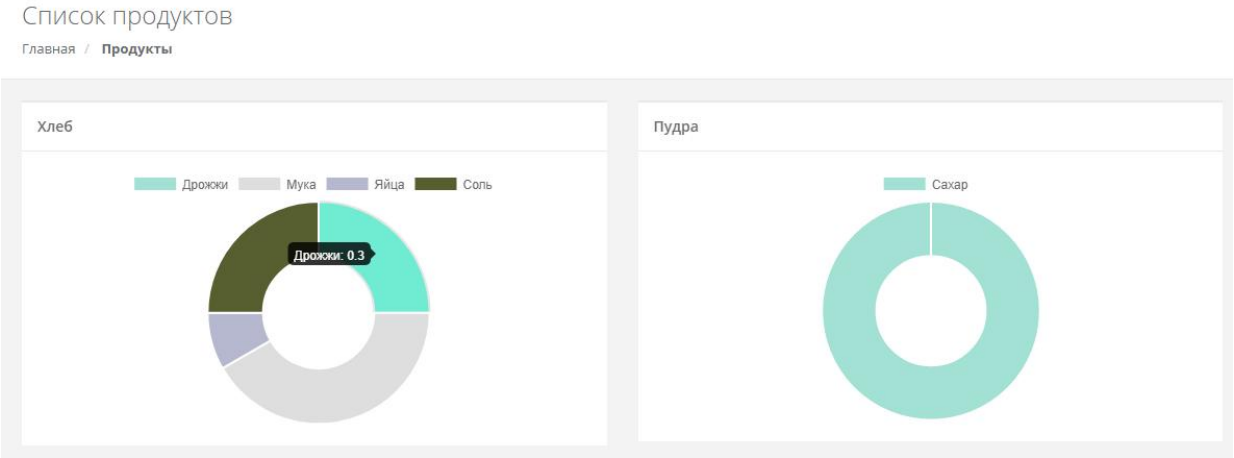


Рисунок 7.18 – Список продуктів

7.2.5 Розділ «Автомобілі»

В даному розділі виводиться інформація про водіїв, які працюють в пекарні, вони здійснюють доставку готової продукції. Для створення необхідно заповнити форму, поті водії будуть відображаться на сторінці списка водіїв (рис. 7.19).

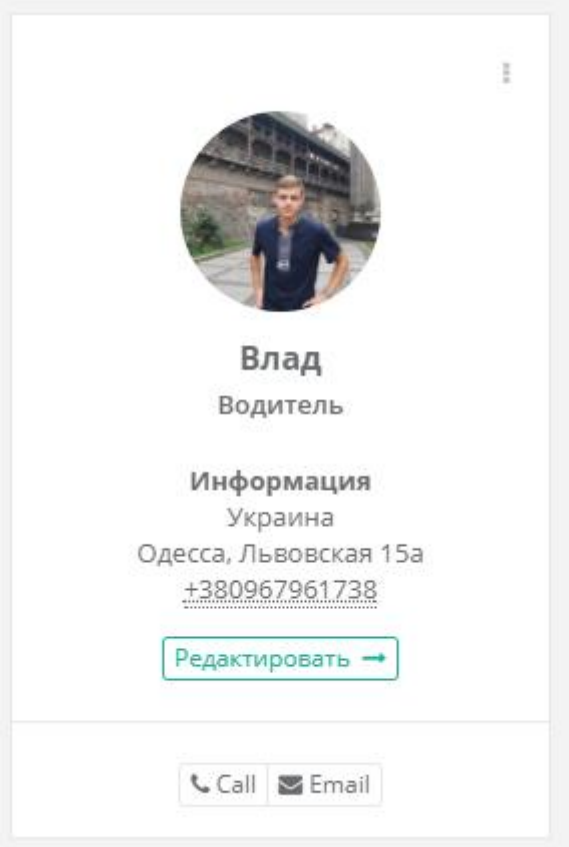


Рисунок 7.19 – Інформація про водія

На сторінці виводиться інформація про водія – фото, його ім’я, адресу проживання, номер телефону та електрону адресу. Також існує кнопка «Редактировать» котра необхідна для редагування інформації про водія. Є можливість натиснути на два посилання «Call» та «Email», якщо по першому то відкриється программа skype з пропозицією здійснити дзвінок по заданому телефону, а якщо натиснути подругому посиланню, то відкриється програма windows outlook, яка запропонує відправити повідомлення на електрону адресу водія.

Адміністратор системи може видалити водія з системи, перейшовши по посиланню «Удалить».

7.2.6 Розділ «Клієнти»

Розділ клієнти необхідний для реєстрації клієнтів котрі здійснюють закази у пекарні. В системі призначене розбиття на юридичних та фізичних осіб. Для цього, в формі існує поле «ИНН» – ідентифікаційний номер платника податків. Довжина в данному полі може бути або 10, або 12 символів.

Приклад створення клієнту представлен рис. 7.20.

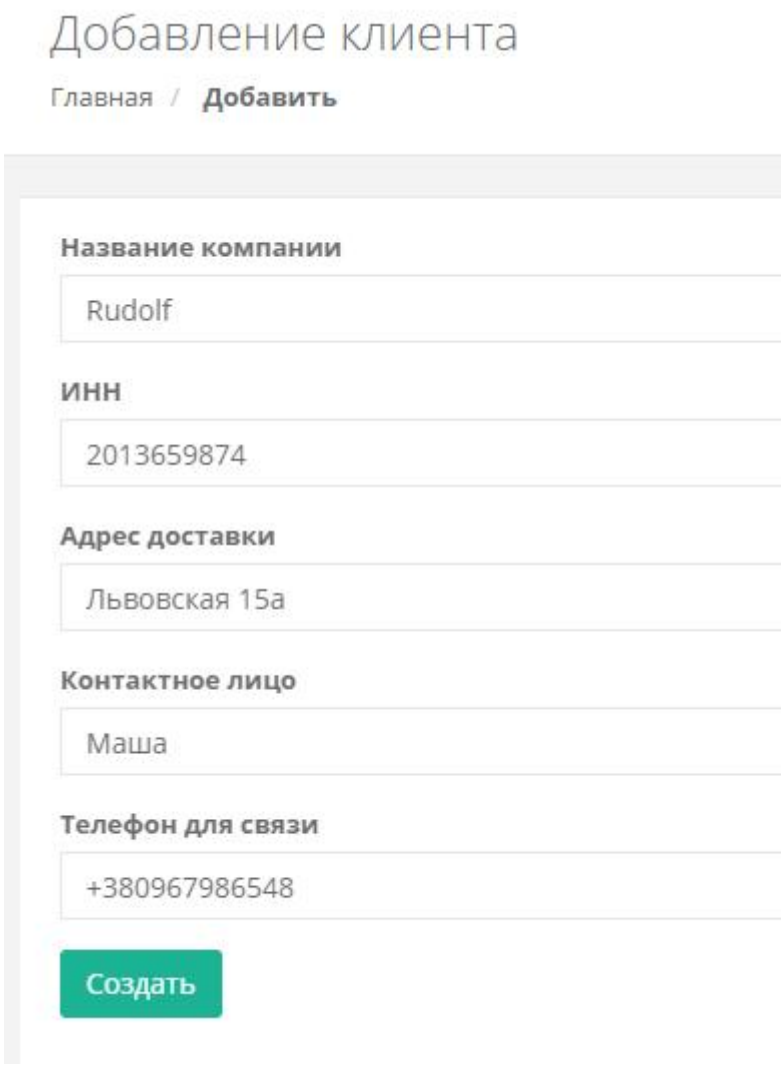


Рисунок 7.20 – Створення клієнта

Форма містить наступні поля: назва компании, ідентифікаційний номер платника податків, адресу доставки, ім’я, номер телефону. Після заповненя необхідними данними та натиску кнопки «Создать» відбувається переадресація на сторінку перегляду всіх клієнтів (рис 7.21).

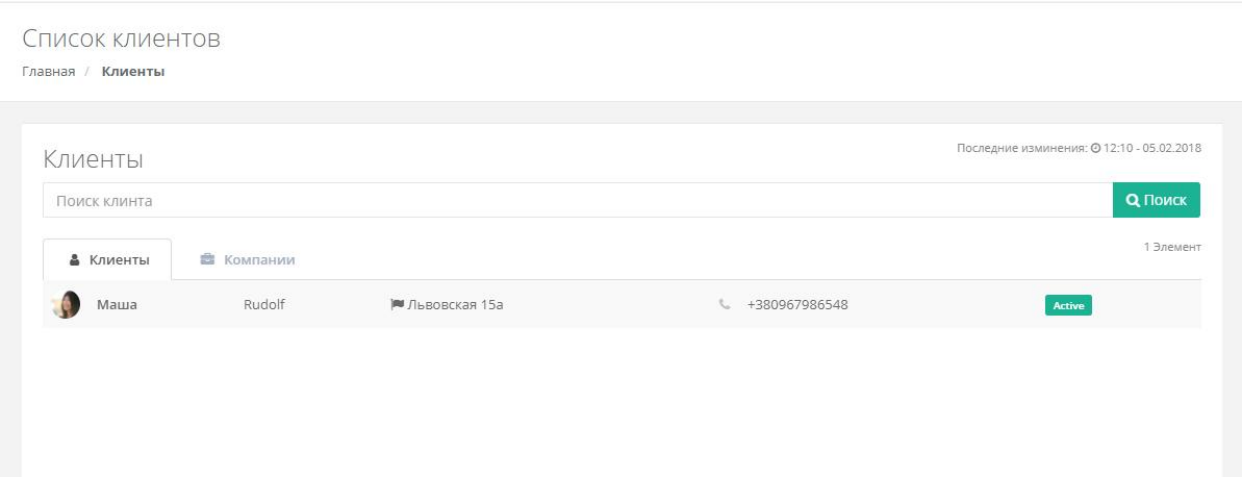


Рисунок 7.21 – Список клієнтів

На данній сторінці присутній форма пошуку, яка здійснює пошук клієнтів за вказаними параметрами, також дату останньої зміни списку клієнтів. Список клієнтів розбитий на дві вкладки, в вкладці клієнти виводиться список всії фізичних осіб, а в вкладці компанії, виводиться список юридичних осіб (7.22).

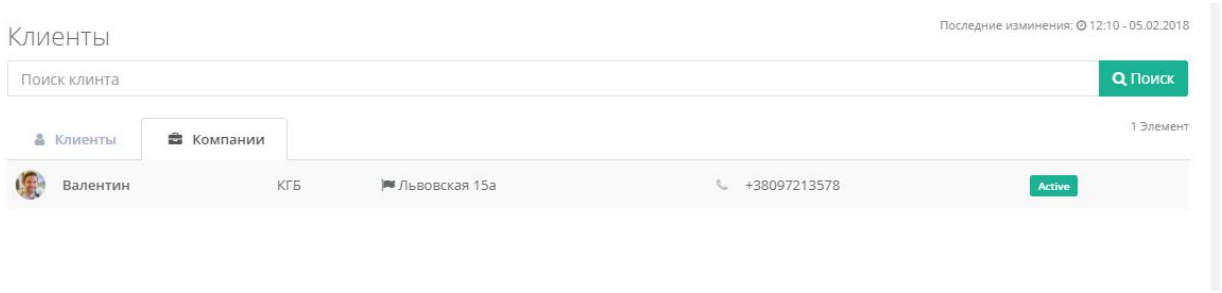


Рисунок 7.22 – Список юридичних осіб

7.2.7 Розділ «Заявки»

У цьому розділі диспетчер системи створює заявки для пекарів. Заявки створюються від бажань клієнтів. Для створення заявки необхідно заповнити поля форми: дата поставки, указати клієнта, якому буде доставлений товар, продукти і в якій кількості. Створення заявки представленно на рис. 7.23.

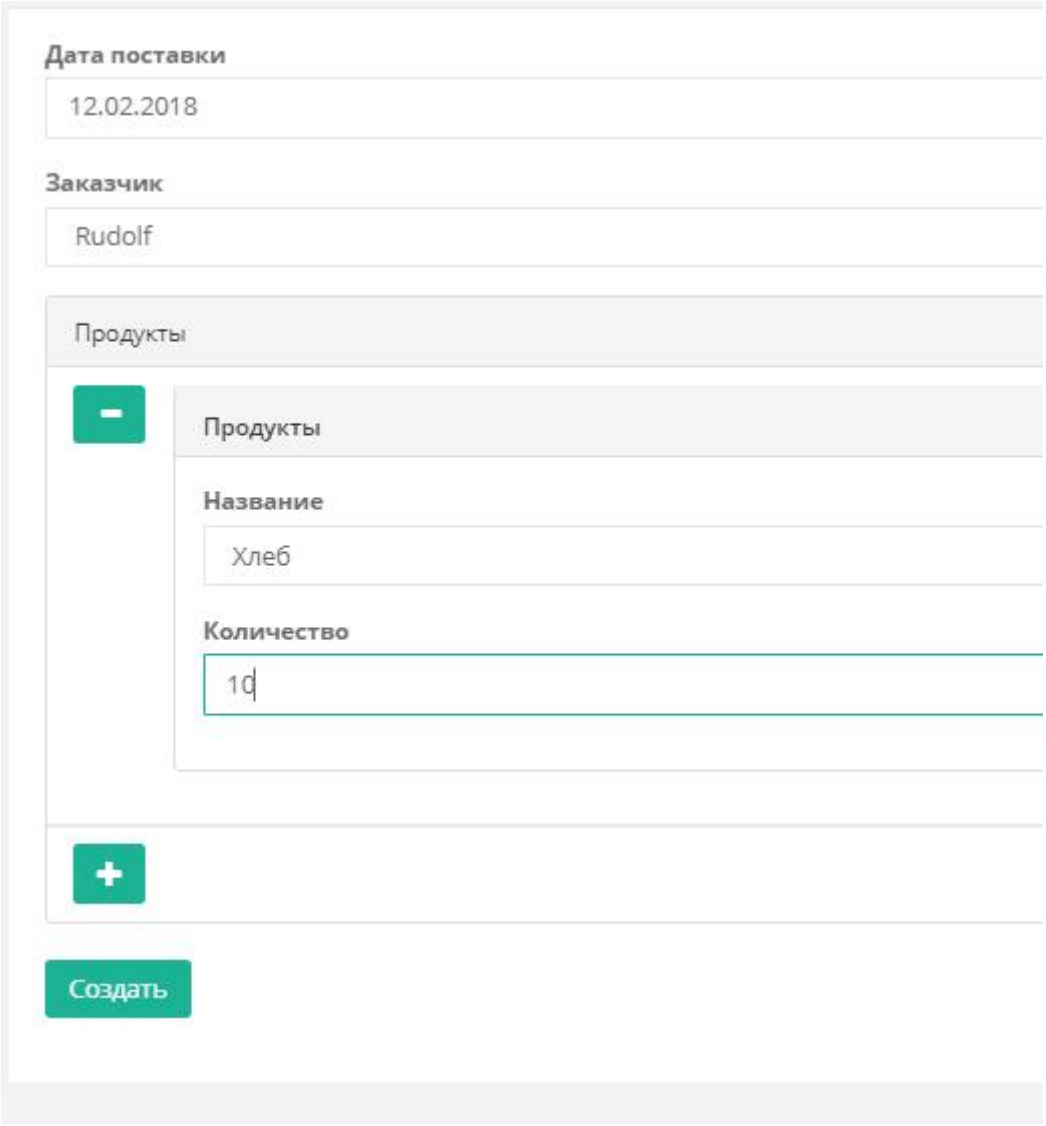


Рисунок 7.23 – Створення заявки

Далі створюється заявка на вибране число в календарі. Якщо перейти в розділ список заявок, і в календарі вибрати відповідну дату, то вона з’явиться в списку (рис. 7.24). Поточну заявку можно редагувати натиснувши на кнопку «Редактировать», з’явиться форма редагування, в якій будуть заповнені всі поля відносно поточної заявки. Це стає можливим завдяки способу – публікації (publications)

База даних програми може мати десятки тисяч записів, деякі з яких можуть бути приватними або вкрай важливими. Зрозуміло, ми не будемо відправляти на клієнт всю нашу базу даних, тому що в більшості випадків це погано відіб'ється на безпеці і продуктивності.

Так що нам потрібен спосіб, за допомогою якого ми зможемо відправляти на клієнт тільки необхідні дані. В данному випадку про всі дані про заявку.

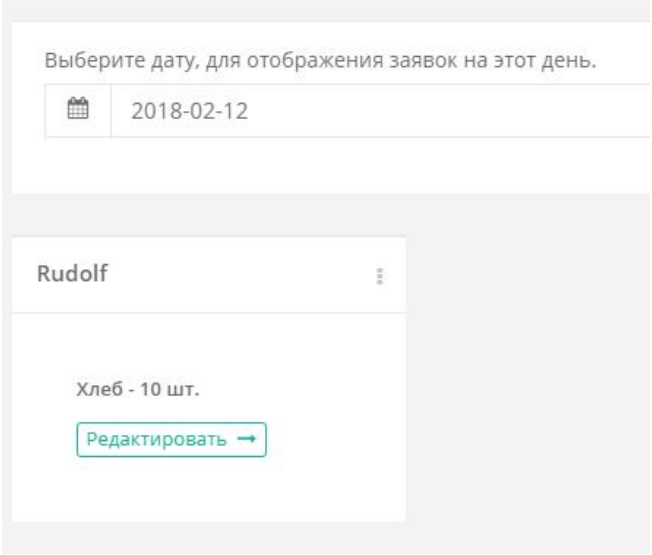


Рисунок 7.24 – Перегляд заявок

7.2.8 Розділ «Завдання пекарям»

Після створених заявок, диспетчер створює завдання для пекарів на основі вказаних данних в заявках. На сторінці завдання пекарям присутній календар з можливістю вибору дати і якщо на вибрану дату існує заявка, тоді всі заявки відобразяться в данному розділі (рис. 7.25).

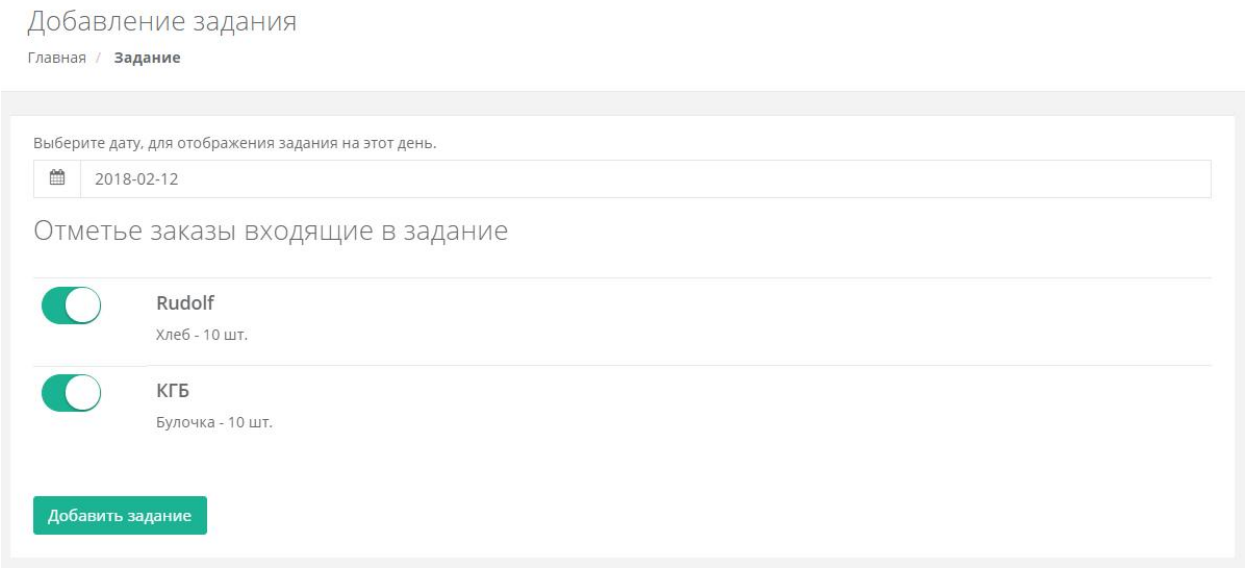


Рисунок 7.25 – Створення завдання для пекарів

Необхідно відмітити ті завдання котрі необхідні і через перемикачі вімкнути необхідні. Натиснувши на кнопку «Добавить задание» завдання буде створенне.

Якщо заявок не існує на вибрану дату, тоді система видасть наступне повідомлення дивись рис. 7.26.

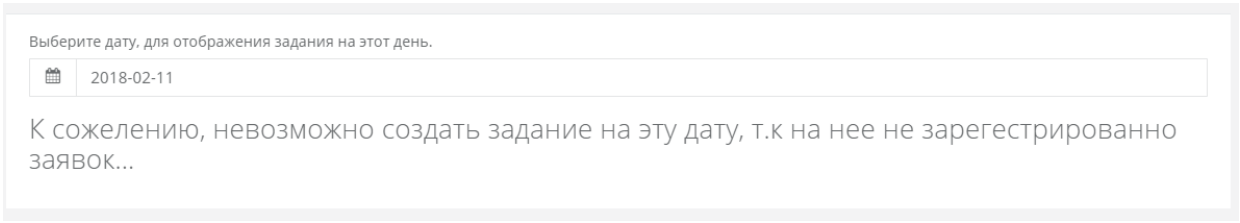


Рисунок 7.26 – Повідомлення при створенні завдання

Всі створенні завдання відображаються на сторінці перегляду всіх завдань (рис. 7.27).

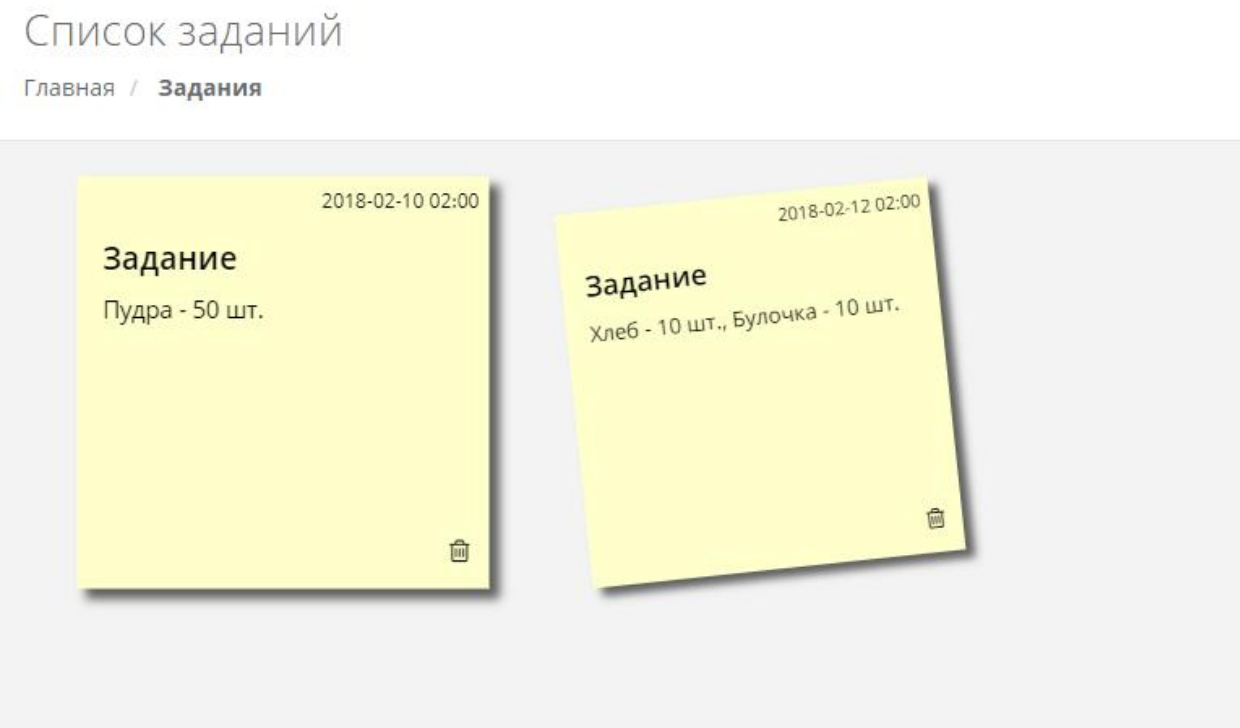


Рисунок 7.27 – Перегляд завдань

7.2.9 Розділ «Аутентифікації»

Для входу в систему необхідно пройти аутентифікацію, заповнивши необхідні поля в формі входу в систему (рис. 7.28).

Для виходу з системи присутнє посилання в вертикальному меню «Выход».

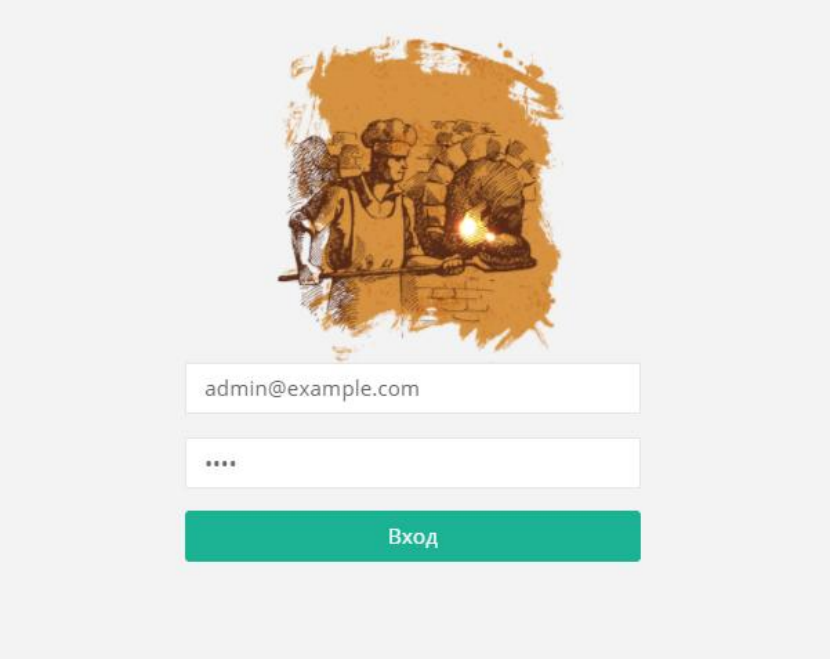


Рисунок 7.28 – Аутентифікація

Для роботи з користувачами був використаний пакет alanning roles. Взагалі кажучи, зазвичай існує дві сторони для будь-якої програми: користувачі, люди, які споживають або використовують наше програмне забезпечення, і адміністратори, люди, які працюють за кадром для надання цього програмного забезпечення. На стороні адміністраторів, як правило, є функціональні можливості, доступ до яких мають лише адміністратори.

Окрім використання методів маршрутизації для контролю доступу користувачів до різних частин нашої програми, ми також маємо варіант використання ролей. Ролі є заздалегідь визначеними групами, які ми можемо призначити користувачам та посиланням у нашій програмі при обмеженні певних частин функціональних можливостей. Наприклад, у нас є деякі посилання в меню котрі повинні використовувати тільки адміністратори, ми можемо сховати кнопку видалення документа з боку звичайних користувачів, але розкривати її для адміністративних користувачів. Використовуючи ролі, ми можемо це досягти.

7.3 Двомовність

Питання інтернаціоналізації призначеного для користувача інтерфейсу одне з важливих питань при розробці програми. Для цього недостатньо використовувати Unicode і перевести на потрібну мову всі повідомлення користувача інтерфейсу. Інтернаціоналізація додатка означає щось більше, ніж під-

тримка Unicode. Дата, час, грошові суми і навіть числа можуть по-різному представлятися на різних мовах.

Широке поширення отримали умовні скорочення термінів інтернаціоналізації та локалізації додатків i18n і l10n, в яких цифра означає кількість символів між першою і останньою позицією:

– i18n – інтернаціоналізація (internationalization);
– l10n – локалізація (localization).

В окремій літературі роблять акцент на цих двох визначеннях, під якими розуміється:

– інтернаціоналізація – це процес розробки програми такої структури, при якій додаток нової мови не вимагає перебудови і перекомпіляції (збірки) всьої програми.
– локалізація передбачає адаптацію інтерфейсу додатку під кілька мов. Додавання нової мови може внести певні складності в локалізацію інтерфейсу.

Для реалізації двомовності був використаний пакет tap:i18n (рис. 7.29). Код реалізації вказан у додатку В.

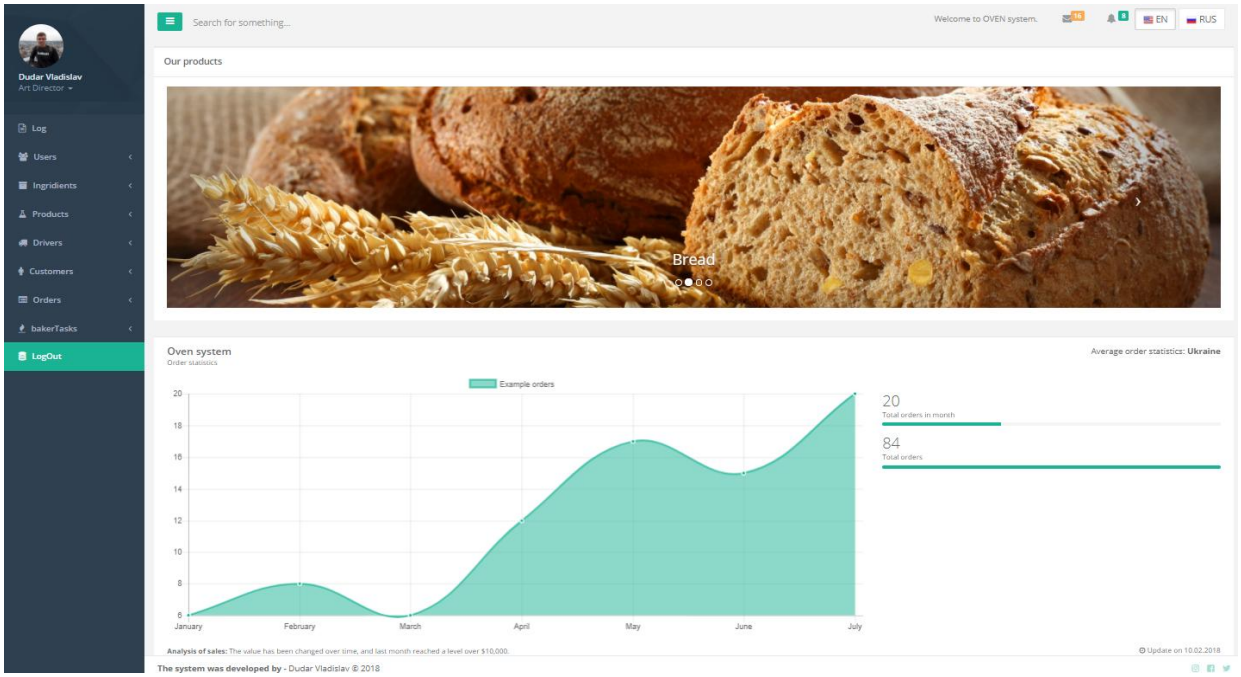


Рисунок 7.29 – Локалізація системи

7.4 Хмарний репозиторій Bitbucket

Складно уявити розробку більш-менш серйозних програмних продуктів без системи управління версіями коду. А для того, щоб проект був успішний, необхідно контролювати роботу програмістів і відслідковувати ефективність в цілому, ставити завдання, вести документацію проекту.

Система управління версіями коду (VCS) дозволяє вести історію змін коду і файлів. Через неї можна проконтролювати і побачити хто вносив зміни в код, коли вносилися зміни і в які частині коду. Також система управління версіями коду дозволяє в короткі терміни відкотитися до потрібної версії коду продукту.

Все більше набирає популярність BitBucket хостинг системи управління версіями, як і GitHub в ньому реалізовані функції VCS з можливістю використовувати GIT і Mercurial як системи управління версіями.

Особливістю BitBucket, в порівнянні з GITHUB – це можливість безкоштовно створювати приватні репозиторії коду (private) в необмеженій кількості, а платити доведеться якщо кількість користувачів по всім вашим проектам буде більш 5.

Єдине обмеження – розмір сховища не повинен перевищувати 2 гігабайти. Хмарний хостинг системи управління версіями коду BitBucket зображений на рисунку 7.30 [17].

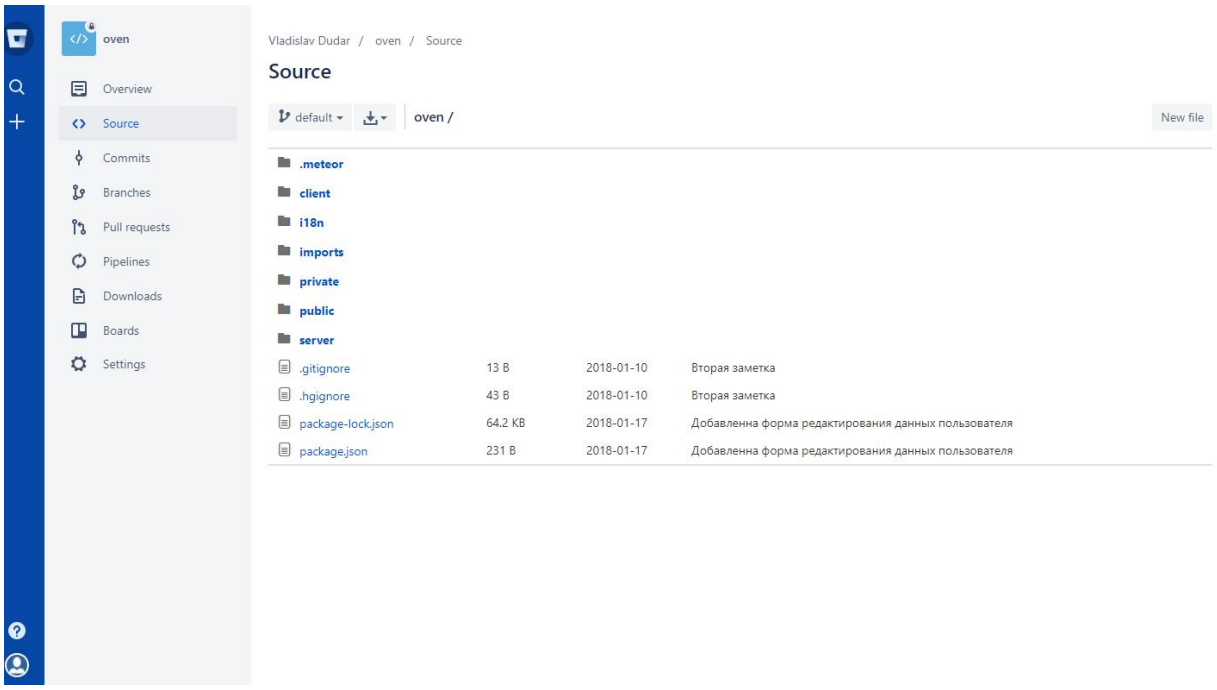


Рисунок 7.30 – Хмарний хостинг системи

ВИСНОВКИ

Створюючи систему управління пекарнею, ставилося завдання навчитися працювати з full-stack технологією Meteor.

Meteor – це передова web-платформа, в якій реалізовано багато цікавих ідей. Її підтримка даних реального часу приваблива і важлива, особливо з урахуванням того, що в інших технологіях підтримка реального часу в кращому випадку запізнюється. Взаємодія в режимі реального часу набуває все більшого значення для майбутнього Web, і здатність Meteor легко і швидко працювати зі складними наборами даних в режимі реального часу також буде ставати все більш цінною.

Команда Meteor постійно працює над розширенням можливостей тих, хто хоче створювати односторінкові розширені клієнтські Web-додатки з високим рівнем інтерактивності. Більшість користувачів віддають таким програмам перевагу перед традиційними багатосторінковими додатками з індивідуальної завантаженням сторінок. Таку архітектуру використовують багато найбільших поштові та офісні сервіси на основі web-інтерфейсу, включаючи Google Gmail, Microsoft Hotmail і Yahoo Mail. Як правило, web-додатки такого роду створюються тільки найбільшими компаніями. Основна місія платформи Meteor полягає в створенні фундаменту, що дозволяє з легкістю будувати додатки такого класу.

В системі управління пекарнею реалізовані механізм аутентифікації користувачів і розмежування прав доступу по ролям, до інформації. Управління пекарнею реалізує всі визначенні вимоги та функціональні можливості для всіх категорій користувачів системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Pal – секрети успішної пекарні, інструкція до дії [Електронный ресурс] – Режим доступу: http://www.pvpak.ru/page/sekrety-uspeshnoy-pekarni-instrukciya-k-deystviyu.
2. Організація хлібопекарні [Електронний ресурс] – Режим доступу: http://uareferat.ga/40f8a-1.html
3. Vipichka – історія міні пекарень в нашій країні [Електронный ресурс] – Режим доступу: http://vipvkusnyashka.ru/kak-poyavilas-pekarnya-istoriya-mini-pekaren-v-nashej-strane.html.
4. IT українською – огляд 5 найпопулярніших JavaScript-фреймворків 2017 [Електронний ресурс] – Режим доступу: http://it-ua.info/news/2017/02/14/oglyad-5-naypopulyarnshih-javascript-freymvorkv-2017.html
5. Вікіпедія – Ангуляр [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/AngularJS
6. Хабрахабр – Переваги використання JavaScript фреймворків [Електронний ресурс] – Режим доступу:https://habrahabr.ru/post/321844/
7. getInstance – основи використання Gulp для збірки JavaScript-додатків [Електронний ресурс] – Режим доступу: https://getinstance.info/articles/tools/introduction-to-gulp/
8. Хакер – atom текстовий редактор [Електронний ресурс] – Режим доступу: https://xakep.ru/2014/02/27/62122/
9. Методи рішення багатокритеріальних задач вибору [Електронний ресурс] – Режим доступу: http://um.co.ua/4/4-18/4-180270.html
10. Студопедия – багатокритерійні завдання і можливі шляхи їхнього рішення [Електронний ресурс] – Режим доступу: http://studopedia.com.ua/bagatokriteriyni-zavdannya-i-mozhlivi-shlyahi-ihnogo-rishennya.html
11. DeveloperWorks – розробка простих web-сайтів реального часу на платформі Meteor [Електронный ресурс] – Режим доступу: https://www.ibm.com/developerworks/ru/library/wa-meteor/index.html.
12. Хабрахабр – як влаштований Meteor зсередини [Електронний ресурс] – Режим доступу: https://habrahabr.ru/post/205878.
13. Medium – розробка з Meteor.js і реактивне програмування [Електронный ресурс] – Режим доступу: https://medium.com/@thought_sync/meteor-js-9863111dfed8.

14. IT українською – розробка Meteor.js і реактивне програмування [Електронний ресурс] – Режим доступу: http://it-ua.info/news/2014/08/11/rozrobka-z-meteorjs--reaktivne-programuvannya.html
15. Фізичне проектування бази даних. [Електронний ресурс] – Режим доступу: http://da.coolreferat.com.ua/nuda/proektuvannya-bagatovimirnih-baz-danih/main.html
16. Bitbucket хмарний GIT репизиторій [Електронний ресурс] – Режим доступу: http://jak.magey.com.ua/articles/bitbucket-krutij-hmarnij-git-repozitorij.html

Д О Д А Т К И

ДОДАТОК А

ДОДАТОК Б

ДОДАТОК В