МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп’ютерних систем

(повна назва кафедри (предметної, циклової комісії))

**Дипломна робота**

на здобуття освітньо-кваліфікаційного рівня «бакалавр»

(освітньо-кваліфікаційний рівень)

на тему Система планування та контролю виконання задач\_\_\_\_\_\_\_\_\_\_\_

на мікропідприємстві\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Planning and task control system for microenterprise\_\_\_\_\_\_\_\_\_\_\_

t

Виконала: студентка денної форми навчання

спеціальності 123 – Комп’ютерна інженерія \_

(шифр і назва напряму підготовки, спеціальності)

Гуменюк Марина Геннадіївна

(прізвище, ім’я, по-батькові)

Керівник Розновець О. І.

(науковий ступінь, вчене звання, прізвище та ініціали, підпис)

Рецензент .

(науковий ступінь, вчене звання, прізвище та ініціали)

|  |  |
| --- | --- |
| Рекомендовано до захисту:  Протокол засідання кафедри  №\_\_\_ від «\_\_\_»\_\_\_\_\_ 2020 р.  Завідувач кафедри  \_\_\_\_\_\_\_\_\_\_\_\_ \_Є.В. Малахов\_  (підпис) (прізвище, ініціали) | Захищено на засіданні ЕК № \_\_\_  протокол № \_\_ від «\_\_\_» \_\_\_\_ 2020 р.  Оцінка / /\_\_\_\_\_  (за національною шкалою, шкалою ECTS, бали)  Голова ЕК  \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) (прізвище, ініціали) |

Одеса – 2020

АНОТАЦІЯ

В дипломній роботі розроблюється тема «Система планування та контролю виконання задач на мікропідприємстві».

Метою дипломної роботи є створення інформаційної системи у вигляді веб-додатку для планування, розподілу та контролю виконання задач на мікропідприємстві. Призначенням розроблюваної системи є автоматизація вищезазначених процесів, що дозволяє керівнику мікропідприємства проводити контроль поточного стану доручених співробітникам завдань і термінів їх виконання, отримувати об'єктивну оцінку продуктивності роботи співробітників, і, як наслідок, забезпечувати ефективне та своєчасне виконання завдань. У якості мікропідприємства розглядається веб-студія.

Превагами створеного додатку є заощадження мікропідприємством фінансових ресурсів на придбання недешевих систем управління проектами, а також економія часу на навчання персоналу за рахунок забезпечення співробітників лише тими інструментами, що їм необхідні.

Інформаційна система створена у стандарті REST за допомогою стеку технологій MERN: СУБД MongoDB, фреймворку ExpressJS, бібліотеки для розробки інтерфейсів ReactJS та платформи розробки веб-додатків NodeJS.

ABSTRACT

АННОТАЦИЯ

Зміст

[ВСТУП 7](#_Toc42573458)

[1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 10](#_Toc42573459)

[2 ВИБІР АРХІТЕКТУРИ ТА ШАБЛОНУ ПРОЕКТУВАННЯ ДЛЯ СТВОРЕННЯ Інформаційної системи 18](#_Toc42573460)

[2.1 Вибір архітектури інформаційної системи 18](#_Toc42573461)

[2.2 Вибір шаблону проектування 19](#_Toc42573462)

[3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ БАЗИ ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПЛАНУВАННЯ ТА КОНТРОЛЮ ВИКОНАННЯ ЗАДАЧ ДЛЯ МІКРОПІДПРИЄМСТВА 22](#_Toc42573463)

[3.1 Функціональні можливості користувачів інформаційної системи 22](#_Toc42573464)

[3.1.1 Функціональні можливості виконавця 24](#_Toc42573465)

[3.1.2 Функціональні можливості HR (рекрутера) 25](#_Toc42573466)

[3.1.3 Функціональні можливості Менеджера 27](#_Toc42573467)

[3.2 Побудова ER-діаграми бази даних 28](#_Toc42573468)

[3.3 Вибір бази даних 34](#_Toc42573469)

[3.4 Створення бази даних 35](#_Toc42573470)

[4 ЗАСОБИ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ 37](#_Toc42573471)

[4.1 Опис обраної мови програмування 37](#_Toc42573472)

[4.2 Стек технологій MERN 37](#_Toc42573473)

[4.3 Засоби для реалізації авторизації та аутентифікації 39](#_Toc42573474)

[4.4 Axios.... 40](#_Toc42573475)

[4.5 SCSS..............................................................................................................40](#_Toc42573476)

[5 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 43](#_Toc42573477)

[5.1 Single Page Application та React Router 43](#_Toc42573478)

[5.2 Реалізація шаблону проектування MVC в інформаційній системі 43](#_Toc42573479)

[5.3 Опис програмного забезпечення інформаційної системи 46](#_Toc42573480)

[5.5 Розгортання інформаційної системи за допомогою хмарних технологій Heroku та Git 61](#_Toc42573481)

[6 РЕАЛІЗАЦІЯ БЕЗПЕКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ 64](#_Toc42573482)

[6.1 Алгоритм шифрування Bcrypt для захисту інформації 64](#_Toc42573483)

[6.2 SSL.................................................................................................................64](#_Toc42573484)

[ВИСНОВКИ 66](#_Toc42573485)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 67](#_Toc42573486)

[ДОДАТОК А 69](#_Toc42573487)

[Додаток Б 73](#_Toc42573488)

ВСТУП

Якщо поспостерігати за ринком, останнім часом з 'явилася тенденція розвитку малого і середнього бізнесу. Але якщо великі корпорації можуть дозволити собі впровадження різних CRM-систем [[1]](#_Customer_Relationship_Management) і використання узагальнених інструментів для планування проектів і внутрішніх процесів в компанії [[2],](#_40_сервисов_для) то у представників малого бізнесу часто немає особливої необхідності використовувати такі громіздкі, складні та, як правило, вельми недешеві програмні системи. На сьогоднішній день варто враховувати і той факт, що найбільш прибутковою й перспективною є IT-галузь. А це означає, що необхідно зробити все необхідне для того, щоб дати їй можливості розвиватися в економічно складних ринкових умовах.

Останнім часом помічено зростання інтересу підприємців до створення веб-студій у масштабах одного-двох офісів з кількістю працівників до 10 осіб і порівняно невеликим фінансовим оборотом. Згідно класифікації підприємств [[3],](#_Микропредприятия,_малые,_средние) подібні компанії підпадають під категорію мікропідприємств. Веб-студії займаються розробкою веб-сайтів для фізичних та юридичних осіб. В основному, вони виконують приватні замовлення або ж знаходять їх на фріланс-ресурсах. Але для того щоб керувати процесами в такій компанії, співробітникам доводиться вивчати дорогі і складні системи управління проектами, які часто мають надлишковий функціонал, або ж взагалі обмежуватися записками в блокноті.

Таким чином, у керівника мікропідприємства постає проблема: як зробити так, щоб заощадити кошти на придбанні CRM-системи, зекономити час на навчанні персоналу і дати співробітникам лише ті інструменти, що їм необхідні? З метою задоволення потреб малого бізнесу IT-сфери ринку у рамках даної дипломної роботи розроблятиметься інформаційна система для автоматизації планування, розподілу та контролю виконання задач у веб-студії, призначена для того, щоб:

* заощадити час і ресурси;
* забезпечити контроль стану виконуваних співробітниками завдань;
* забезпечити контроль термінів виконання завдань;
* отримати оцінку продуктивності роботи співробітників.

Розроблювана інформаційна система призначена для різних категорій працівників веб-студії. З допомогою цієї системи вони матимуть змогу маніпулювати даними, виходячи з професійних потреб згідно з виконуваними функціями. Зокрема, менеджер повинен мати можливість створювати та розподіляти задачі на виконавців та відслідковувати стан виконання задач, виконавці повинні звітувати під час роботи над кожною з задач, рекрутер повинен фіксувати результати пошуку нових кандидатів на вакансії та вести календар проведення співбесід, а також оцінювати персонал за професійними та особистими якостями.

Також необхідно реалізувати концепцію системи бонусів та штрафів, яка дозволить менеджеру аналізувати результати роботи підлеглих, а виконавцям – відслідковувати власну успішність та заробляти матеріальні бонуси, які мотивують співробітників якісно і своєчасно виконувати їхню роботу.

Отже, для досягнення зазначеної мети у дипломній роботі необхідно вирішити наступні задачі:

* виконати аналіз предметної області;
* визначити категорії користувачів і сформулювати їх вимоги до створюваної інформаційної системи;
* обрати архітектуру і шаблон проектування створюваної інформаційної системи;
* спроектувати базу даних;
* обрати технології та засоби реалізації інформаційної системи;
* забезпечити захист системи від несанкціонованого доступу і розмежування повноважень з боку різних категорій користувачів;
* розробити інтерфейс для різних категорій користувачів, який дасть можливість користувачам системи ефективно маніпулювати даними предметної області відповідно до їх повноважень;
* забезпечити цілісність і безпеку даних як на рівні БД, так і на рівні клієнтського інтерфейсу.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Перед проектуванням інформаційної системи необхідно проаналізувати бізнес-процеси, які мають місце при розробці програмного забезпечення у веб-студії, тобто необхідно окреслити основні схеми взаємодій користувачів системи і власне самої системи. Опис бізнес-процесів проводиться за допомогою діаграм бізнес-процесів методології ARIS [[4].](#_BPM_Tutorials_with)

Над кожним проектом у веб-студії працюють кілька співробітників, об'єднаних в команди. Складність може представляти процес комунікації між ними. Нерідко для обміну інформацією, результатами роботи і документами розробники використовують електронну пошту, месенджери і соціальні мережі. Часто документи пошкоджуються при пересиланні, губляться або відправляються неактуальні версії, що веде до втрати часу на усунення даних неприємностей. Розроблювана інформаційна система повинна спрощувати цей процес: кожному члену команди повинен надаватися особистий кабінет, де, крім відстеження власних завдань і термінів їх виконання, співробітник може прикріплювати доступні іншим членам команди посилання на актуальні результати своєї роботи і створені ним документи, розміщені на зовнішніх ресурсах.

Керівником веб-студії є менеджер. Проектована інформаційна система повинна надати йому можливість створення проектів, розбиття їх на завдання, надавати інструмент, за допомогою якого можна розподіляти завдання серед співробітників, стежити за термінами виконання і за прогресом кожного проекту. Необхідно також реалізувати інструмент для підтримки системи заохочень, за допомогою якої менеджер зможе оцінити кожну задачу в певну кількість балів та обмежити термін її виконання, що дозволить йому мотивувати співробітників веб-студії швидше і якісніше виконувати роботу (у разі недотримання терміну виконання задачі застосовується система штрафів).

Процес розробки програмного продукту у веб-студії виглядає наступним чином. Спочатку приймається замовлення від клієнта, далі воно обговорюється, визначаються технічні та матеріальні питання та створюється технічне завдання. Менеджер при отриманні замовлення ділить проект на певну кількість завдань, визначає виконавців і терміни виконання. Завдання виконуються послідовно. Цей бізнес-процес зображений на рисунку 1.1.

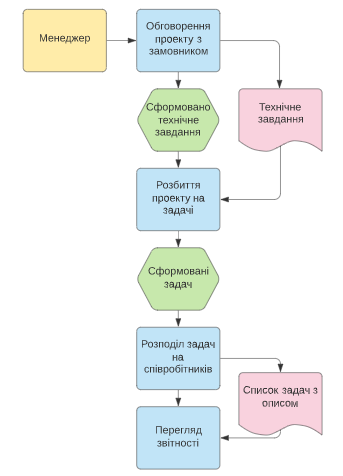


Рисунок 1.1 – Бізнес-процес отримання проекту від замовника

Першим завдання отримує маркетолог. У його задачі входить дослідження цільової аудиторії, проведення соціальних опитувань. Отримані дані маркетолог фіксує у документі хмарного сховища та прикріплює до поточного завдання посилання до звітного документу.

Далі завдання отримує копірайтер: у нього є доступ до звіту з виконаної маркетологом роботи. На підставі цього звіту копірайтер пише текст для сайту і фіксує його також в документі хмарного сховища, прикріплюючи до поточного завдання посилання з результатами своєї роботи.

Використовуючи дані маркетолога та копірайтера, дизайнером за допомогою онлайн-інструментів (наприклад, figma.com) створюється дизайн-макет сайту. Дизайнер прикріплює посилання на виконаний ним проект до поточного завдання.

Таким же чином веб-розробник отримує своє завдання вже з макетом від дизайнера, виконує верстку та прикріплює до поточного завдання посилання на віддалений репозиторій (або на хост, в залежності від поставленої задачі).

Бізнес-процес розробки веб-сайту зображений на рисунку 1.2.

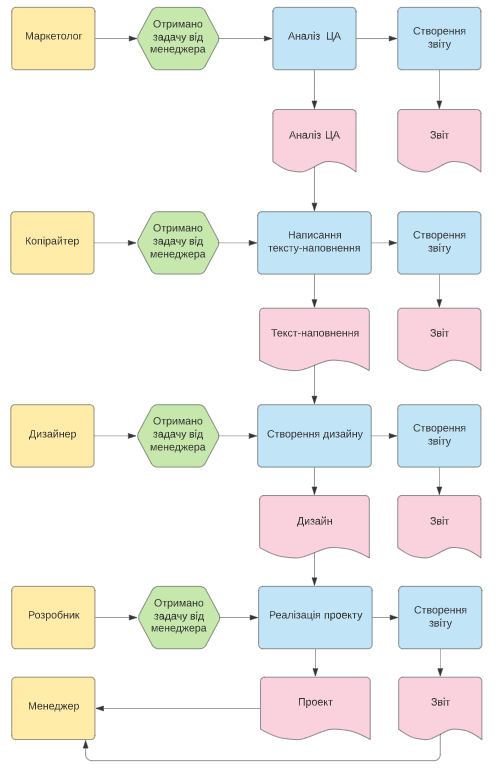


Рисунок 1.2 – Бізнес-процес розробки веб-сайту виконавцями

Робота над проектом контролюється менеджером і HR (рекрутером).

HR виконує дві основні функції. По-перше, він займається підбором нового персоналу: відшукує кандидатів та створює резюме тих з них, які найбільше задовольняють потреби компанії. Після цього HR призначає співбесіду, проводить її, і якщо кандидат успішно пройшов випробовування, його резюме вивчається менеджером, проводиться технічна співбесіда та приймається рішення щодо прийняття кандидата на роботу. Дані бізнес-процеси зображені на рисунках 1.3 і 1.4.

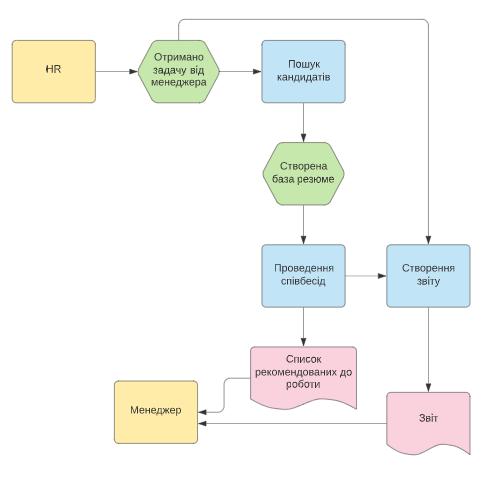


Рисунок 1.3 – Бізнес-процес пошуку потенційного кандидата та проведення співбесіди HR (рекрутером)

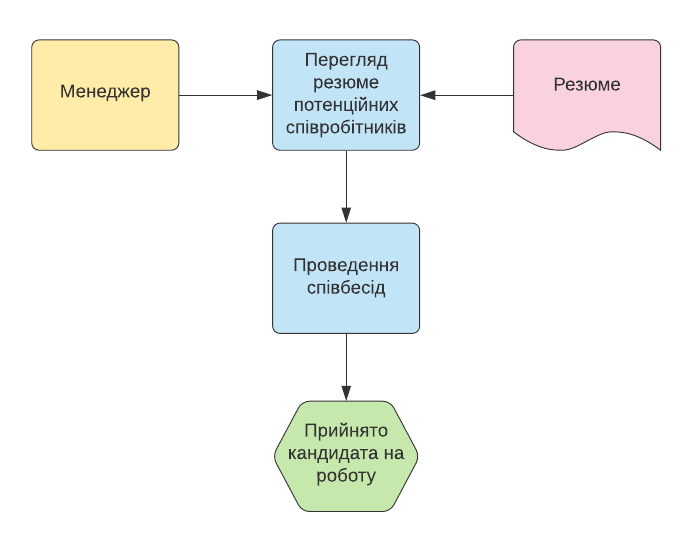


Рисунок 1.4 – Бізнес-процес проведення співбесіди та прийняття на роботу кандидата менеджером

Другим основним бізнес-процесом рекрутера являється оцінка персоналу. Визначаються фактори, які вважає за потрібне оцінити рекрутер, оцінки персоналу періодично занотовуються. Це дозволяє в майбутньому визначити слабкі сторони кожного співробітника в робочому процесі, з метою його вдосконалення. Цей бізнес-процес зображений на рисунку 1.5.

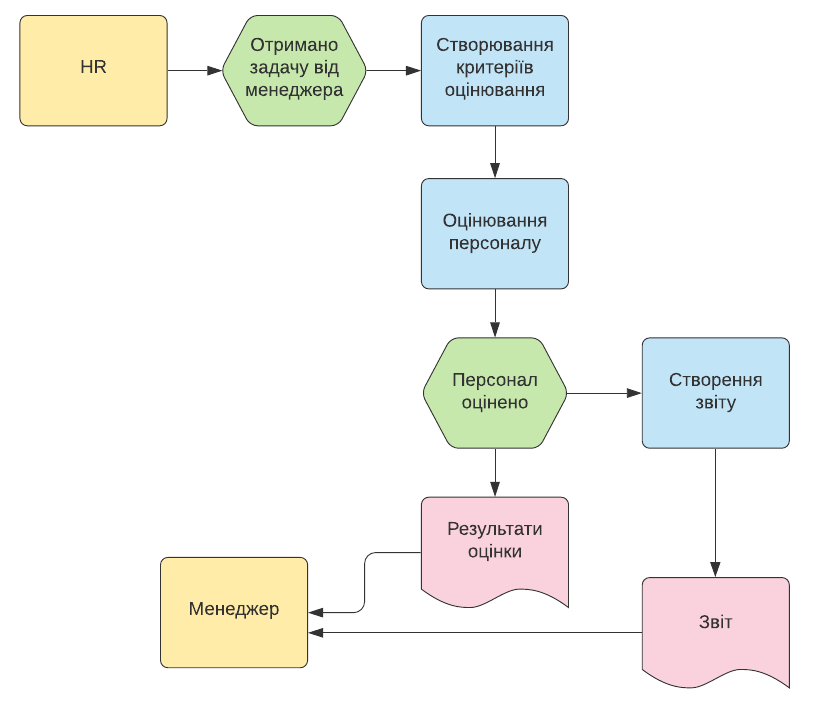


Рисунок 1.5 – Бізнес-процес оцінки персоналу рекрутером

Менеджер повинен мати можливість управляти системою заохочень і штрафів: завдання оцінюється в деяку кількість балів залежно від її складності. Співробітник може виконати завдання, витративши або означений менеджером повний термін (кількість днів), або менше, або більше. При цьому кількість бонусів пропорційно ділиться на кількість днів, відведених на виконання завдання менеджером, тобто:

1. якщо співробітник виконує завдання у визначений термін, він отримує зазначену менеджером кількість бонусів;
2. якщо співробітник виконує задачу раніше, то додатково до зазначених бонусів він має можливість отримати бонуси, які розраховуються за формулою (1.1)

де Б – кількість заздалегідь встановлених за виконання завдання бонусів;

Д – кількість встановлених днів на виконання завдання (термін виконання);

– кількість по факту витрачених співробітником днів на виконання поставленої задачі;

1. якщо співробітник виконує задачу після встановленого терміну, тобто > Д, то отримане значення являється штрафом.

Для того, щоб працювати з розроблюваною інформаційною системою, кожен співробітник веб-студії повинен мати свій обліковий запис. Реєстрацію всіх користувачів та видачу їх паролів також бере на себе менеджер.

Отже, огляд бізнес-процесів, які відбуваються при розробці програмного забезпечення у веб-студії, дозволяє стверджувати, що створювана інформаційна система повинна виконувати наступні функції:

1. спрощувати процедуру планування та розподілу завдань;
2. систематизувати і спрощувати процес розробки веб-сайтів;
3. вирішувати проблему комунікації між членами команди, які працюють над одним проектом;
4. дозволяти контролювати стан робіт та проектів в цілому;
5. ґрунтуючись на особистих та професійних оцінках персоналу, які виставляються рекрутером, давати можливість регулювати настрій в колективі, що дозволить підвищити працездатність працівників;
6. зберігати і надавати доступ до інформації по минулих проектах;
7. здійснювати мотивацію співробітників за допомогою системи бонусів і штрафів.

# 2 ВИБІР АРХІТЕКТУРИ ТА ШАБЛОНУ ПРОЕКТУВАННЯ ДЛЯ СТВОРЕННЯ Інформаційної системи

## 2.1 Вибір архітектури інформаційної системи

Суть архітектури клієнт-сервер досить проста: багато клієнтів (віддалені пристрої) надсилають запити та отримують відповіді від централізованого сервера (хост-комп'ютера). Клієнтські комп’ютери надають інтерфейс, що дозволяє користувачеві комп'ютера робити запити на сервер та відображати результати, які сервер повертає. Сервери чекають надходження запитів від клієнтів, а потім відповідають на них. В ідеалі, сервер забезпечує стандартизований прозорий інтерфейс для клієнтів, щоб клієнти не були в курсі специфіки системи (тобто апаратного та програмного забезпечення), що надає послугу. Клієнти часто знаходяться на робочих станціях або на персональних комп’ютерах, а сервери розташовані в інших місцях мережі, як правило, на більш потужних машинах.

Серед переваг такої архітектури можна виділити:

­­- Централізація контролю: доступ, ресурси та цілісність даних контролюються спеціалізованим сервером, щоб програма чи несанкціонований клієнт не могли пошкодити систему. Ця централізація також полегшує завдання оновлення даних.

- Масштабованість: дається можливість збільшувати потужність клієнтів та серверів. Будь-який елемент можна будь-коли збільшити (або покращити), або можна додати до мережі нові вузли.

- Просте обслуговування: через розподілену роботу серверів та клієнтів при проблемах на якомусь клієнті або сервері робота всієї системи не зупиниться, а тимчасові зміни вплинуть з мінімальною шкодою.

Для реалізації даної системи було прийняте рішення використати трирівневу архітектуру клієнт-сервер, що означає використання наступних компонентів: клієнтський застосунок, підключений до сервера застосунків, який у свою чергу підключений до сервера бази даних (рисунок 2.1.1) . Для цієї архітектури передбачається розмежування рівнів програмного забезпечення на рівень призначеного для користувача інтерфейсу, рівень реалізації прикладних алгоритмів і засобів обробки даних, а також рівень реалізації баз даних, при цьому кожен з них може функціонувати на різних платформах.

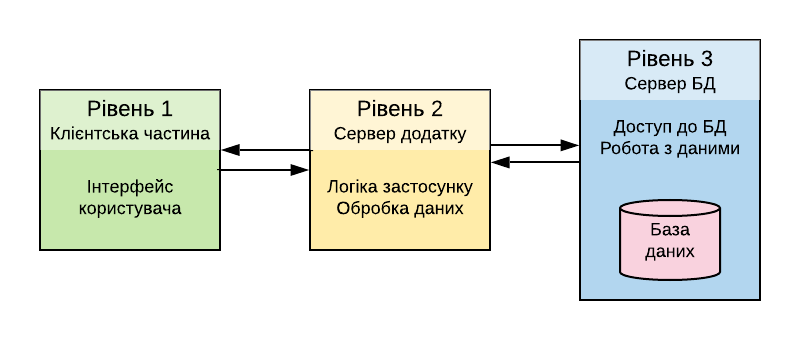


Рисунок 2.1.1 – Трирівнева архітектура клієнт-сервер

Тобто можна сказати, що дана архітектура ідеально підходить для вирішення поставленої задачі, так як серед бізнес-процесів є ті, що виконуються паралельно і не повинні бути обмеженими кількістю та швидкодією пристроїв всередині системи.

## 2.2 Вибір шаблону проектування

Для реалізації інформаційної системи найбільше підходить архітектура MVC (модель – представлення – контролер). Це пов’язано з тим що в MVC контролер відповідає за визначення того, яке уявлення відображати у відповідь на будь-яку дію, в тому числі при завантаженні програми. Це відрізняється від MVP, де дії спрямовуються через подання до доповідача (Presenter). У MVC кожна дія в поданні співвідноситься з викликом контролера і дією. У мережі кожна дія включає в себе звернення до URL-адресою, на іншій стороні якого є контролер, який відповідає. Як тільки цей Контролер завершить свою обробку, він поверне правильний вид. Ще одне велике відміну від MVC полягає в тому, що уявлення безпосередньо не пов'язується з моделлю. Подання просто візуалізується і повністю не має стану.

Одною з найбільших переваг MVC є зменшення порогу входження в проект. MVC поділяє функціональність на три функціональні сфери, забезпечуючи загальну структуру для розробки програмного забезпечення. MVC відокремлює кожну з трьох областей відповідальності, як наслідок даний шаблон проектування являється універсальним та легко підтримується більшістю мов програмування. Розглянемо детальніше кожну з них.

Архітектура MVC полягає в організації коду у вигляді моделей, подань, і контролерів. Моделі підтримують стан додатку, подання піклуються про відображення стану додатку, а контролери забезпечують такі необхідні функціональності для обробки дій користувача і виконання бізнес-логіки, яка змінює стан додатку. Традиційно, в веб-додатку архітектура MVC була реалізована на стороні сервера, щоб тримати логіку подання окремо від бізнес-логіки. Зазвичай, код серверної сторони структурований у вигляді контролера перегляду моделі отримати вигоду зі структури, щоб зробити додаток більш підтримуваним, дозволити можливість паралельної розробки, приведення додатку у відповідність з цілями Progressive Enhancement, і так далі. В цьому випадку код подання піклується про відправку HTML, CSS, і JavaScript до браузеру як частина відповіді для того, щоб показати бажане подання користувачеві. Браузер виводить HTML для відображення вмісту за допомогою HTML тегів, використовує CSS для стилізації вмісту, і використовує JavaScript для додавання інтерактивності подання і організації його поведінки. Це можна побачити на наступній діаграмі, де повна MVC реалізується на на стороні сервера, в той час як на стороні браузера використовуються JavaScript, CSS і HTML (рисунок 2.2.1)

Реалізація даної архітектури робить серверний код більш чистим. Однак код на стороні браузера залишається дуже заплутаним з HTML, CSS, і JavaScript. Це множиться, коли велика кількість коду написана для запуску всередині браузера, для відображення виду і обробки взаємодій. Необхідність запуску коду всередині браузера, не сильно відрізняється від того, який запускається на стороні сервера, звідси і принципи розробки (підтримка, можливість повторного використання, і т.д.), які застосовуються до коду на стороні сервера, також застосовуються до коду на стороні браузера.

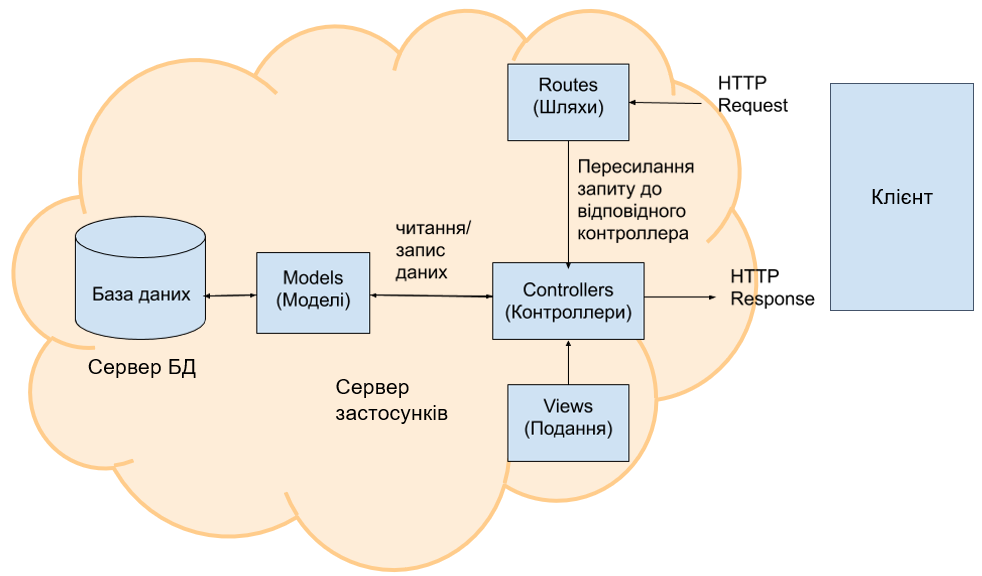


Рисунок 2.2.1 – Схема шаблону проектування Model-View-Controller

# 3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ БАЗИ ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПЛАНУВАННЯ ТА КОНТРОЛЮ ВИКОНАННЯ ЗАДАЧ ДЛЯ МІКРОПІДПРИЄМСТВА

## 3.1 Функціональні можливості користувачів інформаційної системи

Визначення функціональних можливостей користувачів є суттєвою складовою розробки інформаційної системи. Як зазначено вище, для системи планування та контролю виконання задач на мікропідприємстві передбачено декілька категорій користувачів, кожна з яких визначає вимоги по відношенню до системи (тобто визначається, які дії і над якими даними повинен виконувати той чи інший користувач). Набір функцій тієї чи іншої категорії користувачів може відноситися тільки до даної категорії або частково збігатися з набором функцій інших категорій користувачів.

Кожен користувач даної інформаційної системи має свою роль, а ґрунтуючись на ній, має набір доступних функцій. Отже, виділяється 6 категорій користувачів:

1) Менеджер – виконує роль адміністратора і має право перегляду, редагування, видалення інформації про проекти та завдання, а також займається розподілом завдань між користувачами-виконавцями і забезпечує оцінку винагородження за допомогою системи бонусів та штрафів.

2) HR (рекрутер) – користувач інформаційної системи, який займається оцінюванням особистих та професійних якостей співробітників (наприклад, працездатність, товариськість і т. ін.), а також пошуком нового персоналу. Рекрутер має права створення, редагування і видалення власних записів і право перегляду записів, внесених менеджером (завдання до виконання для співробітників, відомості про заохочення).

3) Маркетолог – користувач-виконавець, який має право створення, редагування і видалення власних записів і право перегляду записів, внесених менеджером (завдання до виконання для маркетолога, відомості про заохочення, відомості про проекти).

4) Копірайтер – користувач-виконавець, який має право створення, редагування і видалення власних записів і право перегляду записів, внесених менеджером (завдання до виконання для дизайнера, відомості про заохочення, відомості про проекті), і право перегляду записів, внесених маркетологом (інформація про досліджуваної цільової аудиторії).

5) Дизайнер – користувач-виконавець, який має право створення, редагування і видалення власних записів і право перегляду записів, внесених менеджером (завдання до виконання для дизайнера, відомості про заохочення, відомості про проекти), копірайтером (інформація щодо текстового наповнення сайту) і маркетологом (інформація про досліджувані цільові аудиторії).

6) Веб-розробник – користувач-виконавець, який має право створення, редагування і видалення власних записів і право перегляду записів, внесених менеджером (завдання до виконання для дизайнера, відомості про заохочення, відомості про проекти) і дизайнером (запис про макети з дизайном сайту у вигляді посилань на зовнішні документи).

Визначимо функціональні можливості користувачів розроблюваної системи. Їх можна розділити на 3 групи:

1. Менеджер – зареєстрований користувач з розширеними привілеями;
2. HR (рекрутер) – зареєстрований користувач, що має привілеї виконавця та доступ до спеціальних інструментів (призначення та проведення співбесід, пошук кандидатів, оцінка якостей персоналу);
3. Виконавець – зареєстрований користувач, що має доступ для виконання основного бізнес-процесу (виконання поставлених менеджером завдань).

Функціональні можливості виконавця розроблюваної системи цілком є частиною функціональних можливостей HR (рекрутера), а ті, в свою чергу, ­­– частиною функціональних можливостей менеджера (рис. 3.1).



Рисунок 3.1 – Функціональні можливості користувачів створюваної системи

Далі наведені діаграми функціональних можливостей всіх категорій користувачів створюваної інформаційної системи.

### 3.1.1 Функціональні можливості виконавця

Виконавець – користувач системи, що бере участь безпосередньо в головному бізнес-процесі мікропідприємства – реалізації проекту. Йому доступні такі функції:

1. авторизація: можливість увійти в систему, де він має власний кабінет користувача;
2. перегляд призначених йому задач та бонусів за їх виконання;
3. створення звітів при виконанні назначених йому задач та перегляд вже існуючих звітів.

Діаграма функціональних можливостей Виконавця представлена на рис. 3.2.

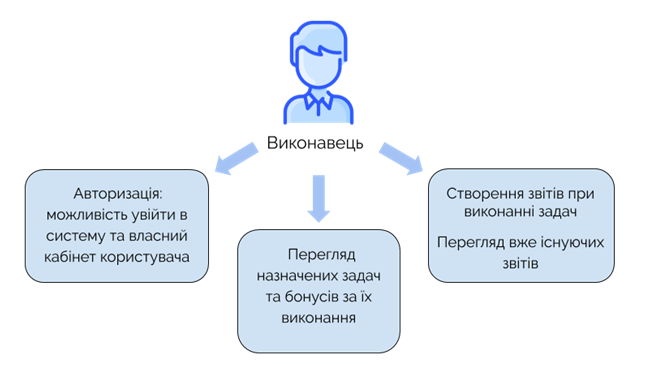


Рисунок 3.2– Діаграма функціональних можливостей Виконавця

### Функціональні можливості HR (рекрутера)

Рекрутер – це зареєстрований користувач інформаційної системи, який авторизувався в ній, має функції звичайного виконавця і функції, що окреслені його бізнес-процесами. Йому доступні такі функції:

1. авторизація: можливість увійти в систему, де він має власний кабінет користувача;

Діаграма функціональних можливостей HR (рекрутера) представлена на рис. 3.3.



Рисунок 3.3 – Діаграма функціональних можливостей HR (рекрутера)

### 3.1.3 Функціональні можливості Менеджера

До всіх перерахованих у п. 3.1.1-3.1.3 функцій для менеджера додається:

1. створення, редагування, видалення будь-якої інформації предметної області;
2. управління користувачами (додавання, видалення, надання ролі);
3. розподіл задач між користувачами та контроль їх виконання;

Діаграма функціональних можливостей Менеджера представлена на рис. 3.4.

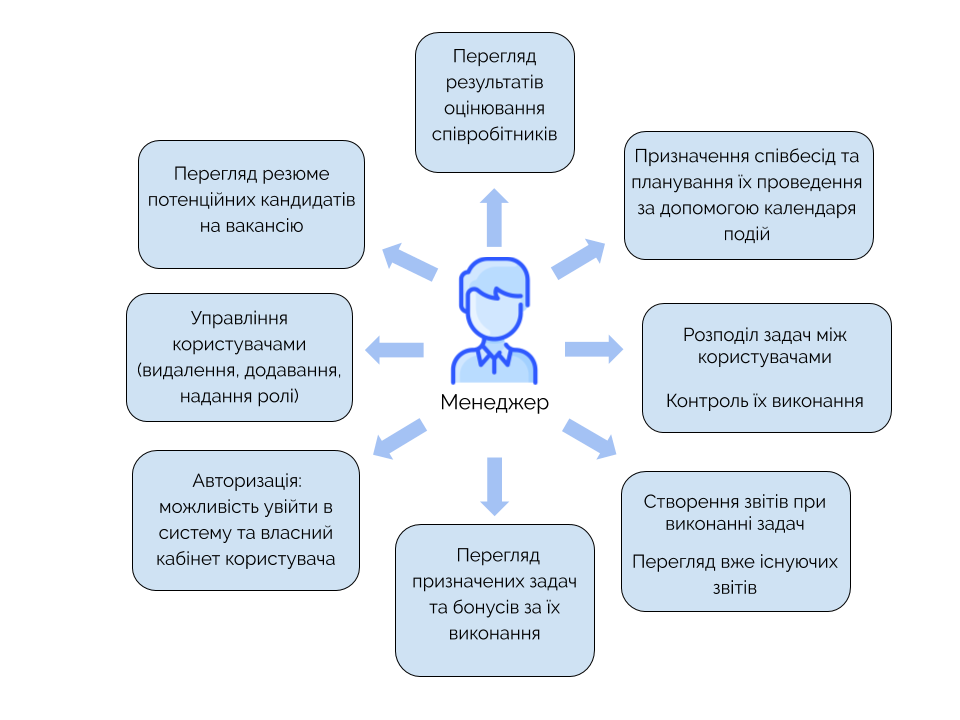


Рисунок 3.4 – Діаграма функціональних можливостей Менеджера

Повний список задач користувачів системи, записаних у форматі користувальницьких історій, а також вхідні та вихідні дані для кожної задачі можна переглянути в Додатку А.

## 3.2 Побудова ER-діаграми бази даних

Під час проектування бази даних необхідно враховувати вимоги до інформаційної системи, визначені при дослідженні предметної області (розділ 1), вимоги користувачів до вхідних та вихідних даних, а також наступні додаткові вимоги:

1. база даних повинна зберігати необхідну інформацію для обліку користувачів інформаційної системи (логін /адреса електронної пошти, пароль);
2. база даних повинна зберігати інформацію про клієнтів веб-студії та їхні замовлення (проекти).

Отже, в БД необхідно зберігати такі відомості:

1. про клієнтів: ім’я, контактний телефон, адреса електронної пошти;
2. про проекти: назва, технічне завдання, термін виконання, виділений бюджет, замовник;
3. про задачі: назва, порядок виконання, бонуси, опис, термін виконання, проект, до якого відноситься задача, виконавець;
4. про звіти: опис (з посиланням на зовнішній ресурс виконаної роботи), дата та час створення та виконання, бонуси, статус роботи, співробітник, що відповідальний за виконання, задача, до якої відноситься звіт;
5. про потенційних кандидатів на вакансію: адреса електронної пошти, контактний телефон, ім’я, пріоритет, коментар щодо кандидата;
6. про працівників: ім’я, посада;
7. про оцінювання стану працівників: співробітник, якого оцінюють, критерій оцінки, оцінка;
8. розклад співбесід (календар): вид співбесіди, кандидат, дата, результат, співробітник, що проводив співбесіду.

Для кожного об'єкта необхідно задати первинні ключі – унікальні ідентифікатори, що однозначно характеризують кожен екземпляр об'єкта, а також унікальні ключі. Для позначення первинних ключів додатково вводимо поля-ідентифікатори. Для об'єктів проект, задача, резюме, звіт, календар, оцінка ідентифікатор об’єкту – первинний ключ. Для об'єкта клієнт та співробітник ідентифікатор об’єктів – первинний ключ, телефон та email – унікальні ключі. Також для користувачів унікальним ключем являється логін для входу в особистий кабінет в даній інформаційній системі.

Таблиця 3.2.1 – Клієнти (Clients)

|  |  |  |  |
| --- | --- | --- | --- |
| Назва стовпця | Призначення поля | Null | Ключі |
| id | Ідентифікатор | Ні | первинний ключ |
| SNP | Ім’я | Ні |  |
| [phone](http://localhost/tools/phpmyadmin/sql.php?db=database&table=user&sql_query=SELECT+%2A+FROM+%60user%60+ORDER+BY+%60user%60.%60password%60+ASC&token=2da73fa5c701ca61e2e76e526d3cce89) | Пароль | Ні | унікальний ключ |
| e-mail | Електронна пошта (e-mail) | Ні | унікальний ключ |

Таблиця 3.2.2 – Проекти (Projects)

|  |  |  |  |
| --- | --- | --- | --- |
| Назва стовпця | Призначення поля | Null | Ключі |
| id | Ідентифікатор | Ні | первинний ключ |
| Name | Ім’я | Ні |  |
| Description | Технічне завдання | Ні |  |
| Deadline | Термін виконання | Ні |  |
| Budget | Бюджет | Ні |  |
| clientId | Ідентифікатор замовника | Ні | унікальний ключ |

Таблиця 3.2.3 – Задачі (Tasks)

|  |  |  |  |
| --- | --- | --- | --- |
| Назва стовпця | Призначення поля | Null | Ключі |
| id | Ідентифікатор | Ні | первинний ключ |
| Name | Назва | Ні |  |
| Order | Порядок | Ні |  |
| Bonuces | Бонуси | Ні |  |
| Description | Опис (технічне завдання) | Ні |  |
| Deadline | Термін виконання | Ні |  |
| projectId | Ідентифікатор проекту, до якого відноситься задача | Ні | зовнішній ключ для зв’язку з таблицею Projects (Проекти) |

Таблиця 3.2.4 – Календар проведення співбесід (Calendar)

|  |  |  |  |
| --- | --- | --- | --- |
| Назва стовпця | Призначення поля | Null | Ключі |
| id | Ідентифікатор | Ні | первинний ключ |
| Type | Тип співбесіди | Ні |  |
| Date | Дата | Ні |  |
| Result | Результат | Ні |  |
| taskId | Ідентифікатор задачі, мета якої – пошук кандидата | Ні | зовнішній ключ для зв’язку з таблицею Tasks (Задачі) |
| reviewId | Ідентифікатор кандидата, з яким назначається співбесіда | Ні | зовнішній ключ для зв’язку з таблицею Reviews (Резюме) |

Таблиця 3.2.5 – Резюме (Reviews)

|  |  |  |  |
| --- | --- | --- | --- |
| Назва стовпця | Призначення поля | Null | Ключі |
| id | Ідентифікатор | Ні | первинний ключ |
| Email | Електронна пошта | Ні | унікальний ключ |
| Phone | Телефон | Ні | унікальний ключ |
| SNP | Ім’я | Ні |  |
| Priority | Пріоритет | Ні |  |
| Comment | Коментар |  |  |

Таблиця 3.2.6 – Звіти (Reports)

|  |  |  |  |
| --- | --- | --- | --- |
| Назва стовпця | Призначення поля | Null | Ключі |
| id | Ідентифікатор | Ні | первинний ключ |
| Description | Опис (з посиланням на зовнішній ресурс з виконаним завданням) | Ні |  |
| Create date | Дата створення | Ні |  |
| Finish date | Дата завершення виконання задачі | Ні |  |
| Bonuces | Призначені бонуси | Ні |  |
| Status | Статус | Ні |  |
| employeeId | Ідентифікатор виконавця | Ні | зовнішній ключ для зв’язку з таблицею Employees (Співробітники) |
| taskId | Задача, до якої створений звіт | Ні | зовнішній ключ для зв’язку з таблицею Tasks (Задачі) |

Таблиця 3.2.7 – Працівники (Employees)

|  |  |  |  |
| --- | --- | --- | --- |
| Назва стовпця | Призначення поля | Null | Ключі |
| id | Ідентифікатор | Ні | первинний ключ |
| SNP | Ім’я | Ні |  |
| Role | Посада працівника | Ні |  |
| Phone | Телефон | Ні |  |
| Status | Статус | Ні |  |

Таблиця 3.2.8 – Оцінки (Grades)

|  |  |  |  |
| --- | --- | --- | --- |
| Назва стовпця | Призначення поля | Null | Ключі |
| id | Ідентифікатор | Ні | первинний ключ |
| criteriaId | Ідентифікатор критерія оцінювання | Ні | зовнішній ключ для зв’язку з таблицею Criterias (Критерії оцінювання) |
| employeeId | Ідентифікатор виконавця | Ні | зовнішній ключ для зв’язку з таблицею Employees (Співробітники) |
| grade | Оцінка | Ні |  |

Таблиця 3.2.9 – Критерії оцінювання (Criterias)

|  |  |  |  |
| --- | --- | --- | --- |
| Назва стовпця | Призначення поля | Null | Ключі |
| id | Ідентифікатор | Ні | первинний ключ |
| Name | Назва | Ні | унікальний ключ |
| Description | Коротка харакретистика | Ні |  |

Структура бази даних у вигляді ER-діаграми зображена на рис. 3.2.10.

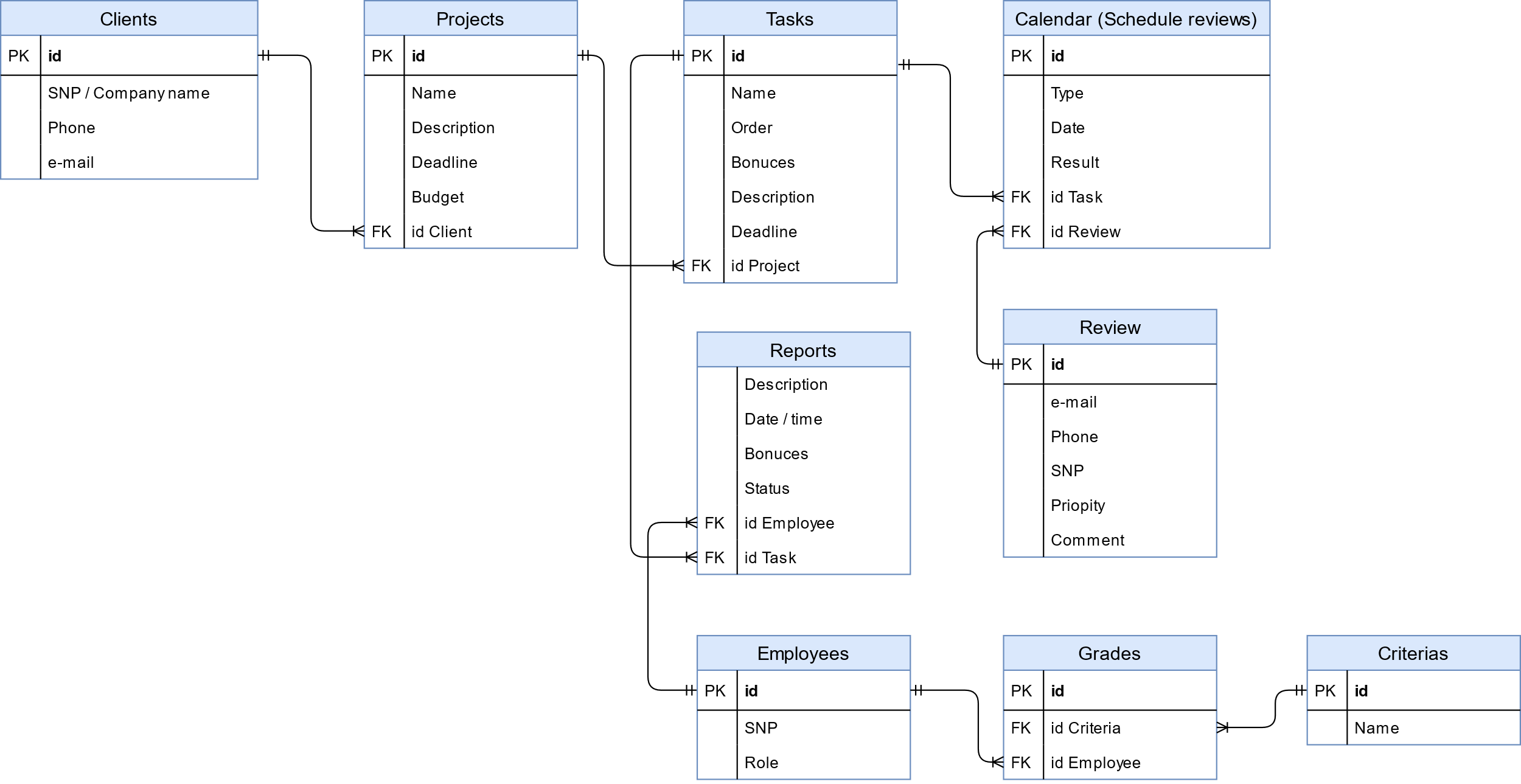


Рисунок 3.2.10 – ER-діаграма бази даних інформаційної системи планування та контролю виконання задач для мікропідприємства

## 3.3 Вибір бази даних

В світі технологій баз даних існує два основних напрямки: SQL і NoSQL, реляційні та нереляційні бази даних. Відмінності між ними полягають в тому, як вони спроектовані, які типи даних підтримують, як зберігають інформацію. Реляційні БД зберігають структуровані дані, які зазвичай представляють об'єкти реального світу. Нереляційні БД зберігають інформацію, наприклад, у вигляді ієрархічних структур даних або об'єктів з довільним набором атрибутів. Те, що в реляційної БД буде розбито на кілька взаємопов'язаних таблиць, в нереляційних може зберігатися у вигляді цілісної сутності.

Дані, з якими працюють співробітники веб-студії, структуровані, при цьому їхня структура не схильна до частих змін. У цьому випадку доцільно не використовувати СУБД, яка дозволяє досить вільно поводитися з типами даних і націлена на обробку великих обсягів інформації, а зберігати дані у реляційних СУБД.

Оскільки NoSQL-СУБД підтримують документоорієнтовану модель даних, то в ній можна зберігати різні документи підприємства (звіти, анкети, тощо) у вигляді сховищ ключів і значень. Також однією з проблем при роботі з будь-якими системами баз даних є збереження даних великого розміру. С ними NoSQL-СУБД при обробці справляються набагато швидше, ніж реляційні. Оскільки в перспективі планується зберігати результати роботи над завданнями (документи) в базі даних, а не на зовнішніх джерелах, кращим вибором тут будуть саме NoSQL-СУБД.

Якщо мікропідприємство буде стрімко зростати, то у цій ситуації також краще буде використовувати NoSQL-СУБД, тому що робота з документами та великими обсягами даних буде проводитися швидше, ніж в реляційній СУБД.

Розроблювана інформаційна система може бути адаптована під будь-який вид інтернет-діяльності. По цій причині доцільніше використовувати NoSQL-СУБД, оскільки в них відсутня жорстка схема бази даних, і легше і швидше відбувається перехід до нової структури БД. У зв'язку зі змінами в предметної області (наприклад, додавання нових посад або посадових обов'язків, зміна роду діяльності підприємства і т. ін.) або зміні концепції зберігання даних (документи замість посилань на зовнішні джерела) немає необхідності наново створювати схему БД.

Ще однією перевагою застосування NoSQL-СУБД є використання хмарних обчислень і сховищ, які вимагають, щоб дані можна було легко розподілити між кількома серверами для забезпечення масштабування. Для тестування і розробки може бути використане локальне устаткування, а потім створена система може бути перенесена в хмару, де вона і буде працювати, і саме для цього створені NoSQL бази даних.

Але слід зауважити, що при всіх перевагах у швидкості і масштабованості обмеження цілісності краще працюють в реляційних СУБД, а в NoSQL-СУБД доведеться ретельно стежити за дотриманням обмежень в програмному коді.

Для реалізації бази даних обрана MongoDB – це NoSQL-СУБД з відкритим кодом, у якій дані зберігаються за допомогою колекцій і документів. Це робить більш легкою та швидкою інтеграцію даних у додатки. MongoDB зберігає дані у двійковому форматі JSON, що дозволяє швидко обмінюватися даними між клієнтом і сервером. MongoDB можна використовувати для зберігання великих обсягів даних, що робить її високо масштабованою.

## 3.4 Створення бази даних

Отримано ряд JSON-об’єктів документоорієнтованої бази даних, які зображені у вигляді таблиць реляційної бази даних на рис. 3.5.

Розглянемо приклад створення однієї з таблиць на прикладі таблиці Criterias. У першому рядку подається назва таблиці:

const Grades = mongoose.Schema(

Далі можна побачити тіло таблиці – оголошується назва кожного рядка та тип даних, що будуть міститись в ньому, а також вказується чи являється дане поле обов’язковим.

  {

    criteriaId: { type: String, required: true },

    employeeId: { type: String, required: true },

    grade: { type: String, required: true },

  },

Після цього вказується чи потрібно зберігати дату та час створення нового об’єкту даного документу:

  {

    timestamps: true,

  }

);

Далі для роботи з даною колекцією необхідно її експортувати:

module.exports = mongoose.model("Grades", Grades);

# 4 ЗАСОБИ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Цей розділ містить відомості про технології, мови та бібліотеки, що застосовуються при розробці програмного забезпечення у даній дипломній роботі.

## 4.1 Опис обраної мови програмування

У якості мови програмування обрана JavaScript – об’єктно-орієнтована мова сценаріїв, що використовується для створення та контролю динамічного вмісту веб-сайту. Для написання даного проекту використовується специфікація ECMAScript 6 [[5],](#_ECMAScript_6_-) що використовується у всесвітній павутині для створення сценаріїв на стороні клієнта, а також застосовується для написання серверних додатків та служб за допомогою Node.js.

## 4.2 Стек технологій MERN

Для розробки інтерфейсу користувача обрана React.js [[11](#_React.js_Документация_[Електронний)] – бібліотека мови програмування JavaScript для створення користувацьких інтерфейсів, які показують динамічні дані. Серед переваг React.js порівняно з іншими бібліотеками, можна виділити те, що ця бібліотека підвищує продуктивність додатків, забезпечує швидку візуалізацію за допомогою віртуального DOM [[6](#_Как_работает_Virtual)], а також гарантує стабільний код за рахунок використання лише спадного потоку даних, при якому зміни, внесені до дитячих структур, не торкаються їх батьків.

Для розробки серверної частини веб-додатку була обрана платформа Node.js [[10](#_Node.js_Docs_[Електронний)] – асинхронний двигун, керований подіями, де додаток робить запит, і потім не зупиняється, очікуючи відповіді, а продовжує працювати над іншими корисними завданнями; по завершенні обробки запиту додаток інформується про результати за допомогою зворотного виклику. Це дозволяє виконувати велику кількість операцій паралельно, що дуже важливо при масштабуванні додатків. Node.js дає змогу писати код на стороні сервера на мові JavaScript без необхідності вивчати іншу мову програмування.

Також у роботі використовується Express JS [[8](#_Веб-разработка_с_применением)] – серверний фреймворк для роботи з Node.js, який допомагає створювати веб-додатки на основі JavaScript.

Вибір СУБД MongoDB обґрунтований у розділі 3.3.

Для роботи з базою даних також використовується Mongoose - це спеціальна ODM-бібліотека (Object Data Modelling) для роботи з MongoDB, яка дозволяє зіставляти об'єкти класів і документи колекцій з бази даних. Після визначення схеми Mongoose дає можливість створити Model (модель), засновану на певній схемі. Потім модель синхронізується з документом MongoDB за допомогою визначення схеми моделі.

Таким чином, у роботі використаний стек MERN [[9]](#_What_is_MERN) (MongoDB, Express JS, React JS та Node JS) – комбінація технологій, заснована на мові JavaScript, що використовується для створення клієнтської та серверної частин сучасних веб-додатків. MERN підтримує шаблон проектування MVC. Схема взаємодії компонентів стеку технологій MERN зображена на рис. 4.2.1.

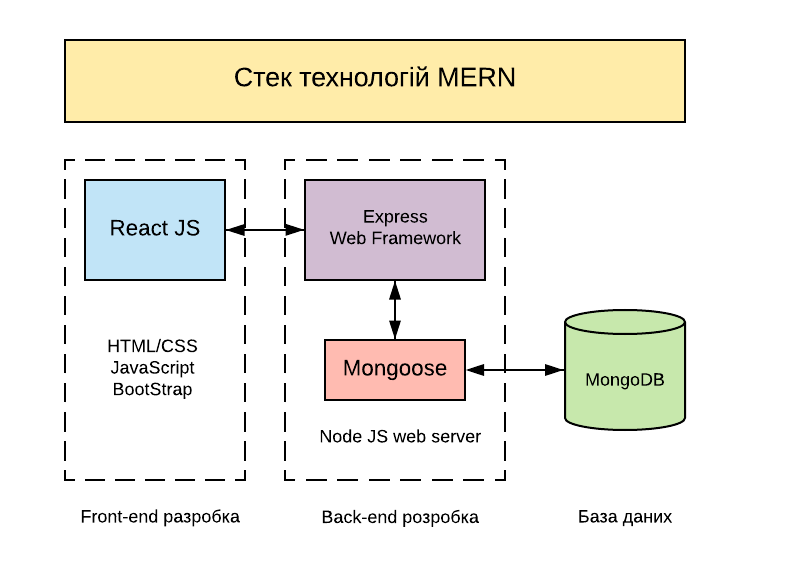


Рисунок 4.2.1 – Схема взаємодії компонентів стеку технологій MERN

## 4.3 Засоби для реалізації авторизації та аутентифікації

JSON Web Token (JWT)[[11]](#_What_is_JSON) - це відкритий стандарт (RFC 7519), який визначає компактний та автономний спосіб безпечної передачі інформації між сторонами як об'єкт JSON. Цю інформацію можна перевірити і довірити, оскільки вона завірена цифровим підписом. JWT можуть бути підписані за допомогою секретного (за допомогою алгоритму HMAC) або пари відкритого / приватного ключів за допомогою RSA або ECDSA.

Основні причини використовувати веб-токени JSON:

1) Реалізація авторизації: після входу користувача кожен наступний запит буде включати JWT, дозволяючи користувачеві отримувати доступ до маршрутів, служб та ресурсів, дозволених за допомогою цього маркера.

2) Підтвердження автентифікації - оскільки JWT можуть бути підписані за допомогою пар відкритих / приватних ключів. Крім того, оскільки підпис обчислюється за допомогою заголовка (header) та корисного навантаження (JWT-claims), це дає переконатись в тому що вміст не був підроблений.

Алгоритм роботи JWT:

1) додаток вимагає авторизації на сервері авторизації;

2) коли авторизація надана, сервер авторизації повертає маркер доступу до програми;

3) додаток використовує маркер доступу для доступу до захищеного ресурсу.

Весь процес можна побачити на рисунку 4.3.1.1.

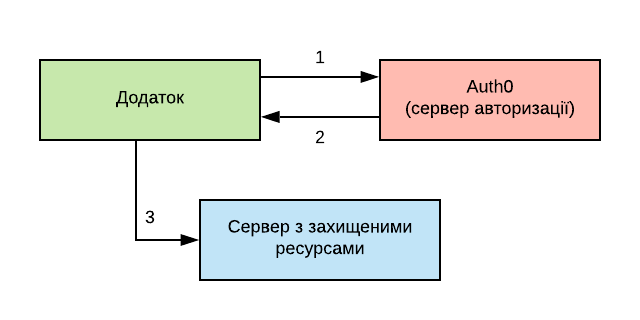


Рисунок 4.3.1 – Алгоритм роботи JWT

Використання JWT має свою низку переваг. JSON при кодуванні має розмір менший ніж XML, що робить JWT більш компактним, ніж SAML.

З точки зору безпеки, SWT може бути симетрично підписаний спільним секретом за допомогою алгоритму HMAC. Однак токени JWT та SAML можуть використовувати пару відкритих / приватних ключів у вигляді сертифіката X.509 для підписання. Підписання XML за допомогою цифрового підпису XML без створення дір у безпеці дуже складно, порівняно підписанням JSON.

JSON-парсери поширені в більшості мов програмування, оскільки вони відображають безпосередньо об'єкти. XML не має природного відображення документа-об'єкта. Це полегшує роботу з JWT порівнюючи з SAML.

JWT використовується в масштабах Інтернету. Це підкреслює простоту обробки клієнтом веб-токена JSON на декількох платформах, особливо мобільних.

## Axios

Axios - це AJAX-бібліотека для виконання HTTP-запитів. Вона була використана через низку переваг:

* працює в середовищі Node.js і в браузерах;
* підтримує використання промісів;
* дозволяє виконувати і скасовувати запити;
* дозволяє задавати тайм-аут відповіді;
* підтримує захист від XSRF-атак;
* дозволяє перехоплювати запити і відповіді;
* підтримує індикацію прогресу вивантаження даних.

## 4.5 SCSS

SCSS [[12]](#_SASS/SCSS_Documentation_-) - це препроцесор, який дозволяє використовувати функції, які ще не є частиною більш широкого стандарту CSS, і забезпечує кращі робочі процеси для підтримки таблиць стилів.

За допомогою препроцесора SCSS можна зменшити кількість повторень і таким чином писати більш читабельний та чистий код.

SCSS повністю сумісний із синтаксисом CSS.

Є декілька причин чому використовувати SCSS для реалізації даної системи краще, ніж звичайний CSS:

1. Вкладеність — SCSS дозволяє вкладати правила CSS один в одного. Вкладені правила застосовуються тільки для елементів, відповідних зовнішніх селекторам.
2. Змінні — в стандартному CSS теж є поняття змінних, але в Sass з ними можна працювати трохи по-іншому. Наприклад, повторювати їх через директиву @for. Або генерувати властивості динамічно. Вдосконалені математичні операції— можна складати, віднімати, множити і ділити значення CSS. На відміну від стандартного CSS, Sass/SCSS дозволяють обійтися без calc().
3. Директиви @for, @while і вираз @if-else — можна писати CSS-код, використовуючи знайомі елементи з інших мов.
4. Міксини (домішки) — можна один раз створити набір CSS-властивостей і працювати з ними повторно або змішувати з іншими значеннями.
5. Функції — можна створювати визначення CSS у вигляді функцій для багаторазового використання.

# 5 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

## 5.1 Single Page Application та React Router

SPA - це програма, яка працює всередині браузера і не потребує перезавантаження сторінок під час використання. Це лише одна веб-сторінка, яку відвідує користувач, на якій потім завантажується весь інший необхідний вміст за допомогою JavaScript.

Вся навігація в системі буде відбуватись на одній веб-сторінці за допомогою бібліотеки React Router. Router визначає набір маршрутів і, коли до додатка, приходить запит, то Router виконує зіставлення запиту з маршрутами. І якщо якийсь маршрут збігається з URL запиту, то цей маршрут вибирається для обробки запиту. Для вибору маршруту також визначено об'єкт Switch. Він дозволяє вибрати маршрут і використовувати його для обробки. Без цього об'єкта Router може використовувати для обробки одного запиту теоретично кілька маршрутів, якщо вони відповідають рядку запиту.

## 5.2 Реалізація шаблону проектування MVC в інформаційній системі

Реалізація шаблону MVC у створюваному проекті включає:

1. реалізацію логіки контролера, що забезпечує основну логіку програми - отримання даних з БД, їх запис, видалення і модифікація;
2. реалізацію логіки моделі для однакового зберігання і модифікації даних, що відносяться логічно до одного програмного елементу;
3. реалізацію подання, не пов'язану з логікою самої програми, а також слабо пов'язаного з типом даних, що працюють з видом - призначеним для користувача інтерфейсом взаємодії, без прив'язаних до нього користувальницьких подій – всі події є прерогативою контролера.

Таким чином, шаблон MVC дозволяє розбити програму на три роздільних модуля, слабо пов'язаних між собою, що дозволяє швидко розширювати, або навпаки скорочувати функціонал програми без модифікації програми в цілому.

Реалізація шаблону проектування MVC в розробленому програмному продукті представлена ​​на рис. 5.1. Під кожним із складових даного шаблону показана його прив'язка до створених класів.

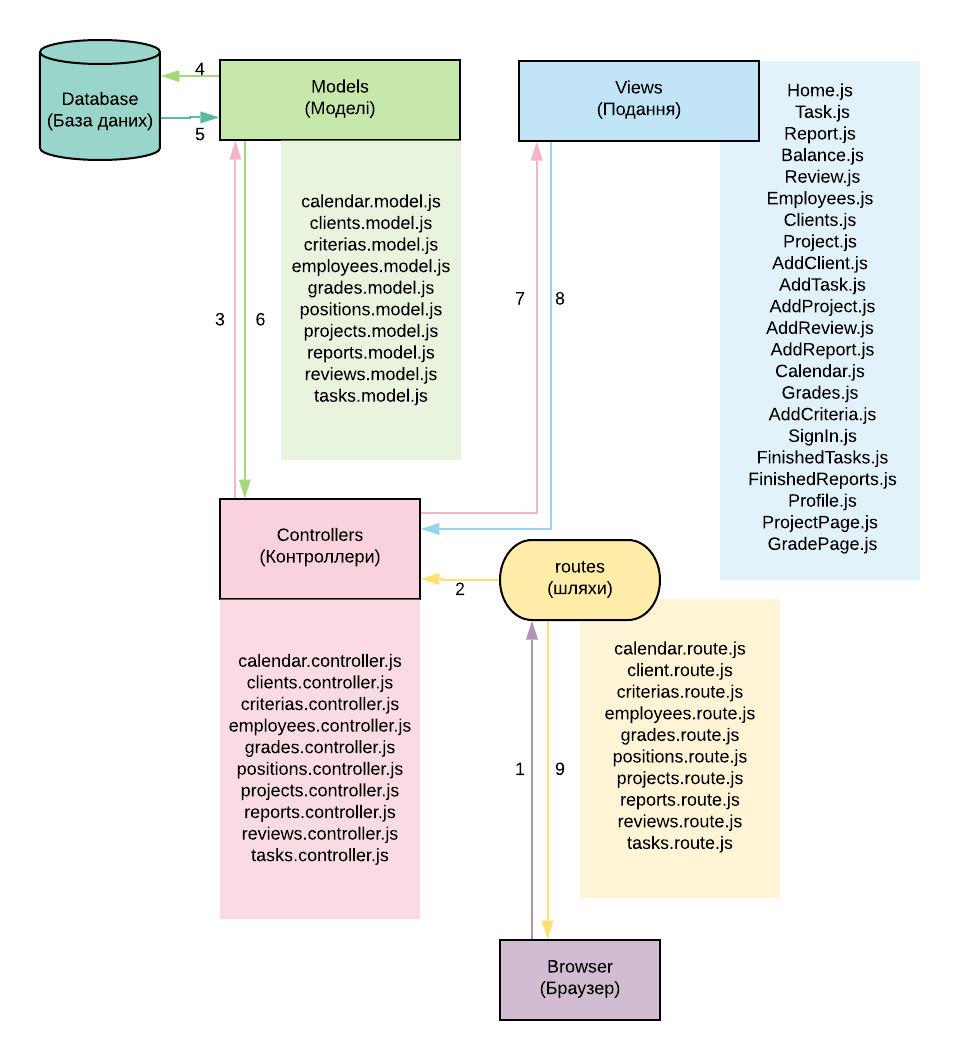


Рисунок 5.2 – Реалізація шаблону MVC

Розглянемо подання, в якому знаходяться такі класи: «Task» необхідний для перегляду існуючих завдань; «Review» необхідний для перегляду резюме потенційних кандидатів на вакантні місця на мікропідприємтсві; «Report» необхідний для створення звітів з виконання задач; «Clients» необхідний для перегляду інформації про клієнтів; «Grades» необхідний для перегляду результатів оцінювання співробітників; «SignIn» необхідний для перегляду та заповнення форми авторизації; «Project» необхідний для перегляду існуючих проектів; «Calendar» необхідний для перегляду запланованих подій (співбесід з потенційними кандидатами на вакансію на мікропідприємстві); «Employees» необхідний для перегляду існуючих співробітників; «AddClient» необхідний для можливості додавати нових клієнтів; «AddTask» необхідний для можливості додавати нових задач; «AddReview» необхідний для можливості додавати нових кандидатів на вакансію (резюме); «AddReport» необхідний для можливості додавати нових звітів; «AddCriteria» необхідний для можливості додавати нових критеріїв для оцінювання; «AddProject» необхідний для можливості додавати нових проектів; «Balance» необхідний для перегляду поточного балансу бонусів та штрафів.

Далі розглянемо структуру контролерів, представлену наступними функціональними компонентами: «Calendar», «Clients», «Criterias», «Employees», «Grades», «Positions», «Projects», «Reports», «Reviews», «Tasks». Вони схожі за структурою, але розділені, бо кожен з них відповідає за маніпуляції своєю об’єктною областю. Роздивимось детальніше основні методи кожного контролера:

1. create – метод для створення об’єкта. Має в собі поля об’єкта, що отримуються з тіла запиту. Також містить в собі проміс save та валідацію valid що перевіряють коректність створюваних даних та зберігають новий документ колекції;
2. findAll – метод для пошуку всіх об’єктів конкретної колекції;
3. findOne – метод для пошуку конкретного об’єкту в колекції;
4. update – метод для оновлення даних об’єкту колекції;
5. delete - метод для видалення даних об’єкту колекції.

Модель представлена класами для роботи з базою даних: «Calendar» - опис моделі запланованих подій (співбесід), яка має поля  employee, review, date, time, name, description;  «Clients» - опис моделі клієнтів мікропідприємства, що має поля phone,  name,    surname,  patronymic,  email; «Criterias» - опис моделі критеріїв для оцінювання, що має поля name, description; «Employees» - опис моделі співробітників мікропідприємства, що має поля name, surname, patronymic, phone, email, password, status,  role; «Grades» - опис моделі оцінок, що має поля criteriaId,  employeeId,  grade; «Projects» - опис моделі проектів, що містить поля name, description, deadline, budget, clientId; «Reports» - опис моделі звітів, що містить поля link, description, date, points,  status,  employeeId,  taskId; «Reviews» - опис моделі резюме потенційних кандидатів на вакансію мікропідприємства, що містить поля name,   surname,   patronymic,  email,   phone,  type,   description,  priority; «Tasks» - опис моделі сформованих задач, що містить поля order,   name,   description, deadline,  bonuce, employeeId.

## 5.3 Опис програмного забезпечення інформаційної системи

Для того щоб користуватись системою, необхідно щоб менеджер надав людині прав користувача. Після цього користувач може зайти під своїм логіном та стандартним паролем в свій профіль. Для цього йому необхідно пройти авторизацію (Рисунок 5.2.1) і ввести свій e-mail та пароль.

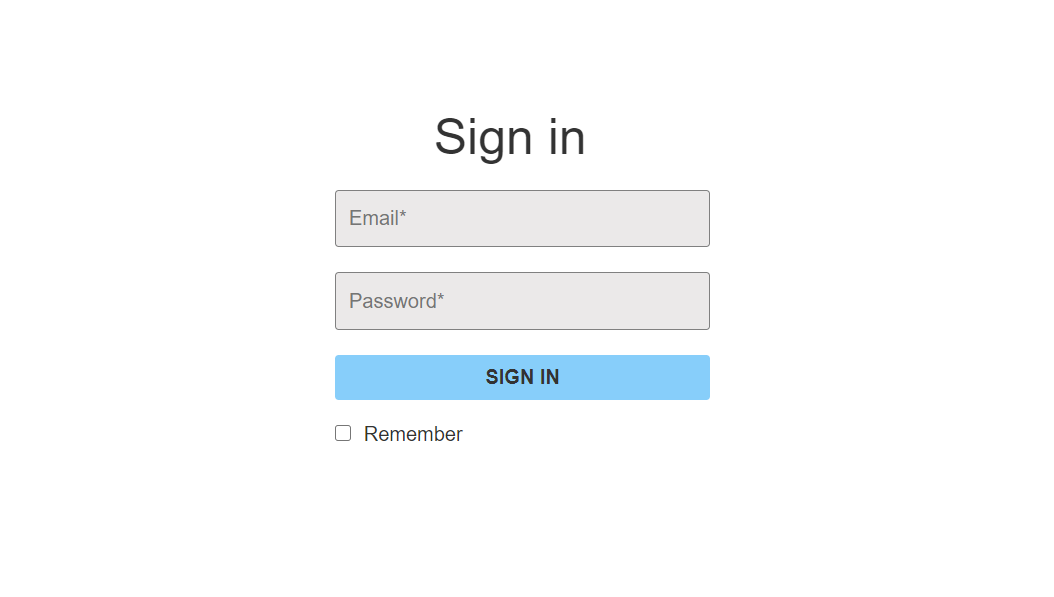


Рисунок 5.3.1 – Авторизація користувача

Даний функціонал оброблює контролер employees.controller.js. При введені користувачем своїх даних для авторизації, вони відправляються на сервер, де проводиться пошук користувача за таким логіном та паролем. Якщо користувач з такими даними існує, тоді генерується JWT-токен і відправляється на клієнт:

exports.auth = async (req, res) => {

try {

const employee = await Employees.findOne({

email: req.body.email,

});

if (

!employee ||

!utils.comparePasswords(req.body.password, employee.password)

) {

throw new Error("Incorrect login/password");

}

const token = employee.generateAuthToken();

res.send({

token,

user: employee.toObject(),

});

} catch (err) {

res.status(404).send({

message: err.message || "Incorrect login/password",

});

}

};

Коли клієнт отримує токен, він його зберігає і використовує при подальших запитах на сервер, додаючи його до заголовку авторизації запиту.

get api() {

const {common} = ApiService.api.defaults.headers;

const token = User.getToken();

if (token) {

common["authorization"] = token;

} else if (common["authorization"]) {

delete common["authorization"];

}

return ApiService.api;

}

Для валідації запиту створена функція проміжної обробки. Якщо токен існує і є коректним, відбувається дешифрування цього токену, і система може визначити ким був зроблений запит. Якщо запит було створено неавторизованим користувачем, його буде відхилено.

module.exports = function (req, res, next) {

const token = req.headers["authorization"];

if (!token) {

return res

.status(401)

.send("Access denied. No token provided.");

}

try {

const decoded = jwt.verify(token,process.env.JWT\_SECRET);

req.user = decoded;

return next();

} catch (ex) {

return res.status(400).send("Invalid token.");

}

};

Якщо авторизація пройшла успішно, людина отримує права доступа згідно своїй ролі. На рисунках 5.3.2-5.3.4 можна побачити екрани різних типів користувачів, залежно від їхньої ролі.

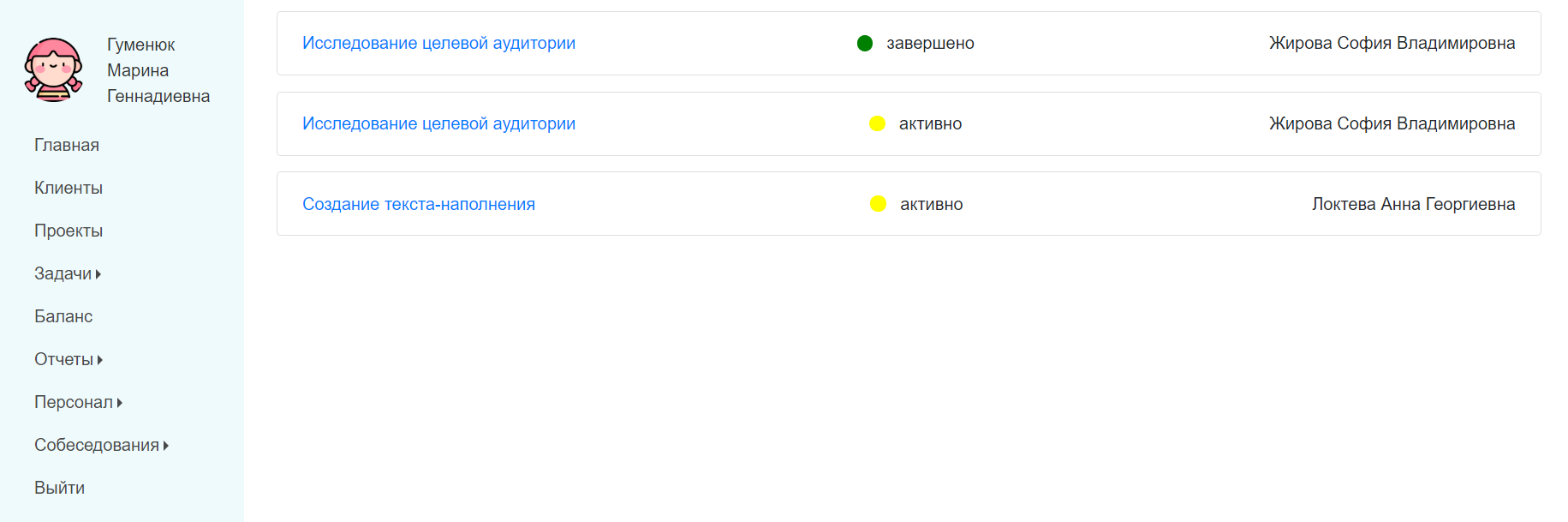


Рисунок 5.3.2 – Профіль користувача «Менеджер»

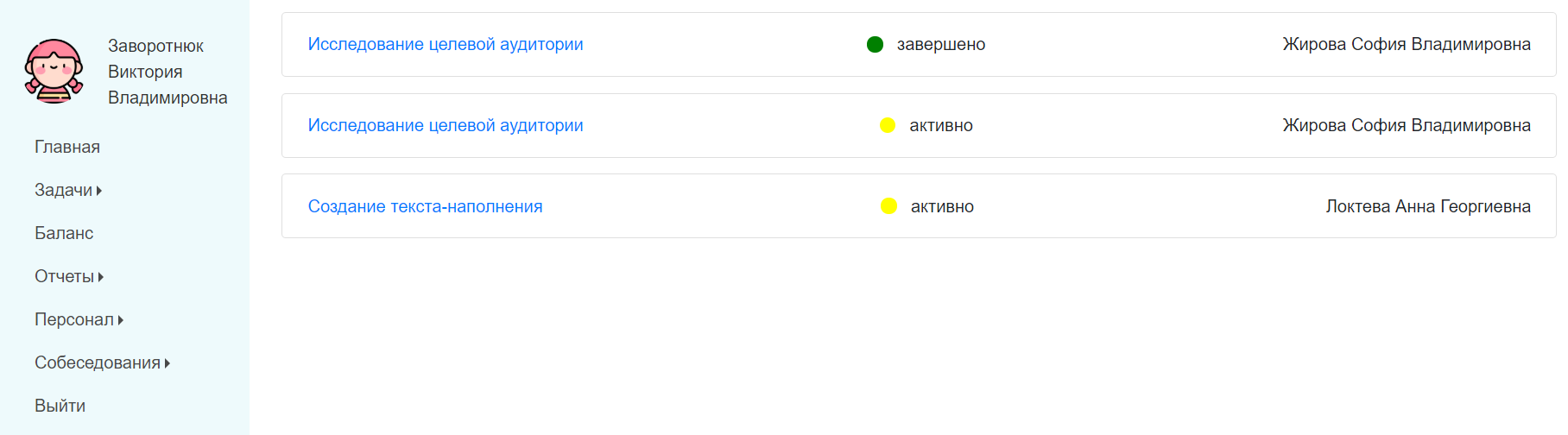


Рисунок 5.3.3 – Профіль користувача «HR (рекрутер)»

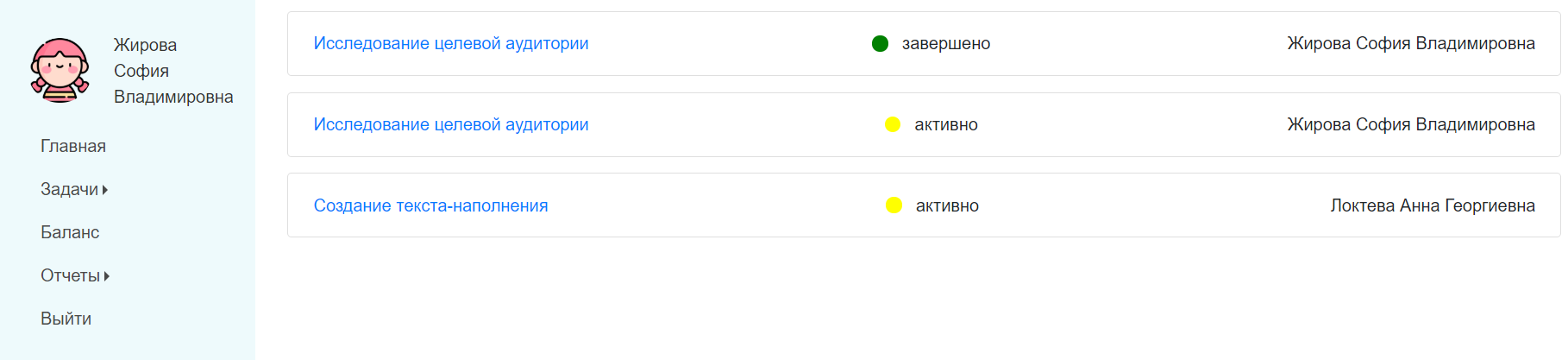


Рисунок 5.3.4 – Профіль користувача «Виконавець»

Менеджер має можливість перегляду інформації про клієнтів, а також їх створення і видалення (рисунок 5.3.5). Цей функціонал обслуговується контролером clients.controller.js, а звернення до бази даних забезпечується за допомогою clients.model.js.

Функція створення отримує дані з Request та зберігає їх у базі даних:

exports.create = (req, res) => {

const client = new Clients({

phone: req.body.phone,

name: req.body.name,

surname: req.body.surname,

patronymic: req.body.patronymic,

email: req.body.email,

});

client

.save()

.then((data) => {

res.send(data);

})

.catch((err) => {

res.status(500).send({

message:

err.message || "Some error occurred while creating the Clients.",

});

});

};

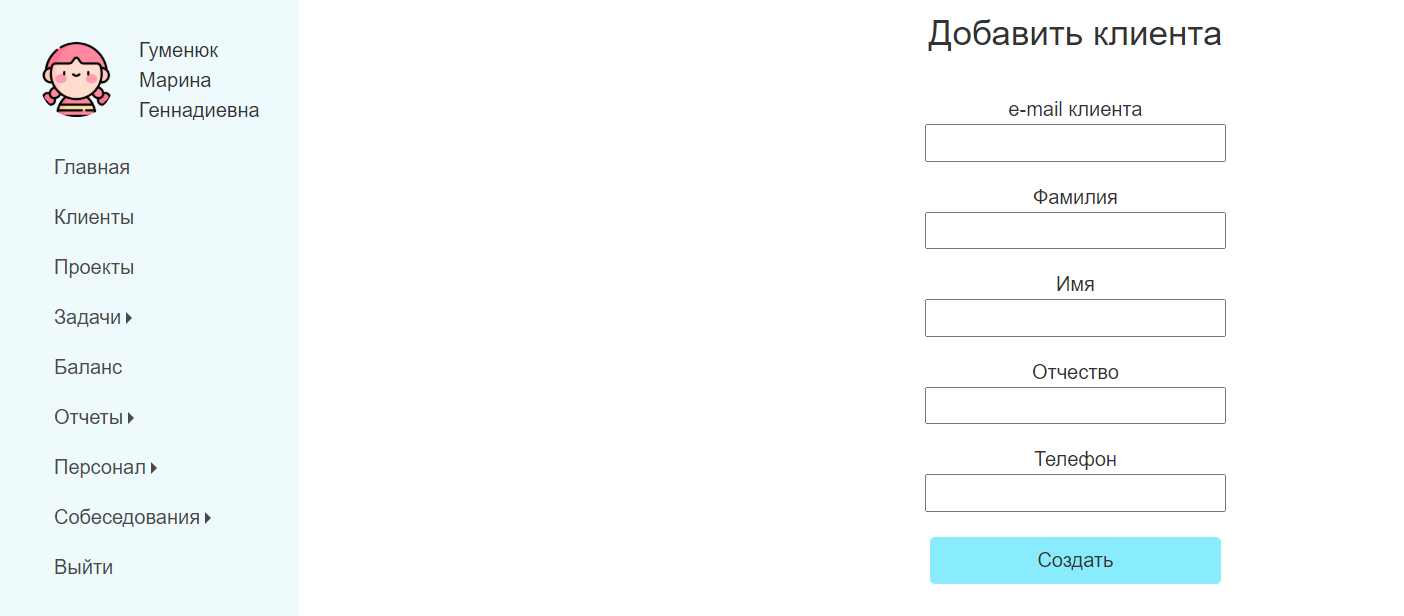


Рисунок 5.3.5 – Створення нового клієнта

Також в цьому контролері є функція видалення, яка з Request отримує конкретного клієнта і видаляє інформацію про нього з бази даних:

exports.delete = (req, res) => {

Clients.findByIdAndRemove(req.params.clientId)

.then((client) => {

if (!client) {

return res.status(404).send({

message: "Clients not found with id " + req.params.clientId,

});

}

res.send({ message: "Clients deleted successfully!" });

})

.catch((err) => {

if (err.kind === "ObjectId" || err.name === "NotFound") {

return res.status(404).send({

message: "Clients not found with id " + req.params.clientId,

});

}

return res.status(500).send({

message: "Could not delete client with id " + req.params.clientId,

});

});

};

І функція що повертає у Response всіх існуючих клієнтів та інформації про них:

exports.findAll = (req, res) => {

Clients.find()

.then((clients) => {

res.send(clients);

})

.catch((err) => {

res.status(500).send({

message: err.message || "Some error occurred while retrieving clients.",

});

});

};

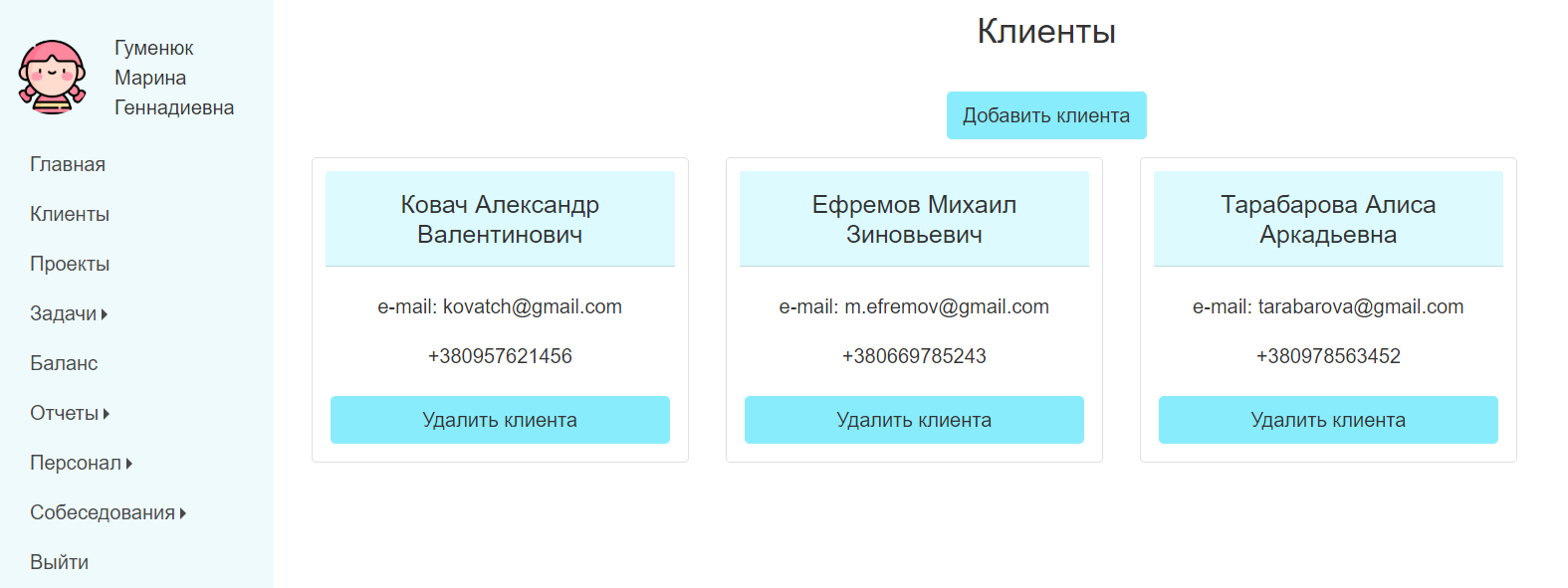


Рисунок 5.3.6 – Інформація про клієнтів

Також менеджеру доступна функція створення та перегляду проектів (рисунок 5.3.7 та 5.3.8). Даний функціонал забезпечується за допомогою контролеру projects.controller.js. Контролери для всіх сутностей дуже схожі між собою, з їхнім кодом можна ознайомитись в Додатку Б. Кожен проект має можливість детального перегляду: можна побачити всі звіти всіх користувачів, що стосуються саме цього проекту (рисунок 5.3.9).

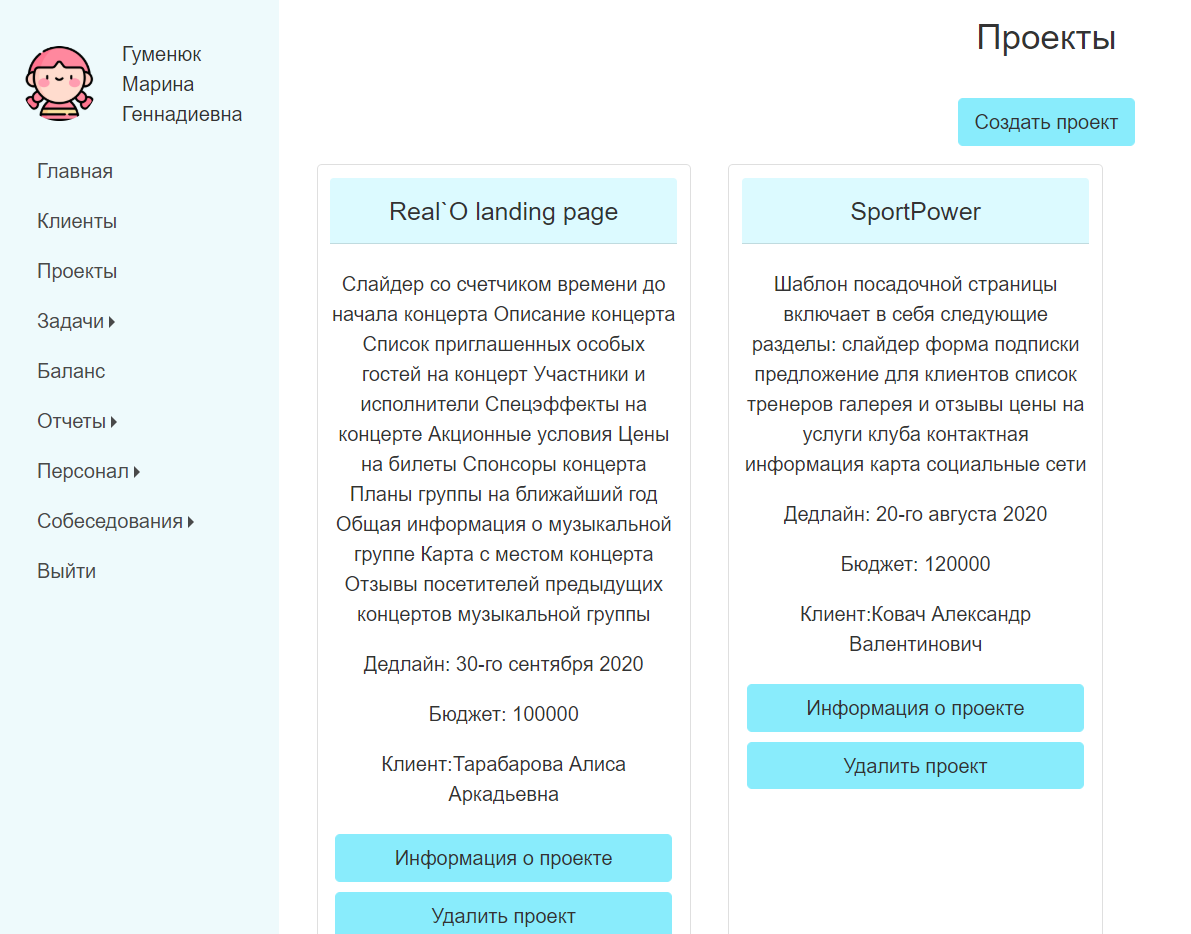


Рисунок 5.3.7 – Перегляд існуючих проектів

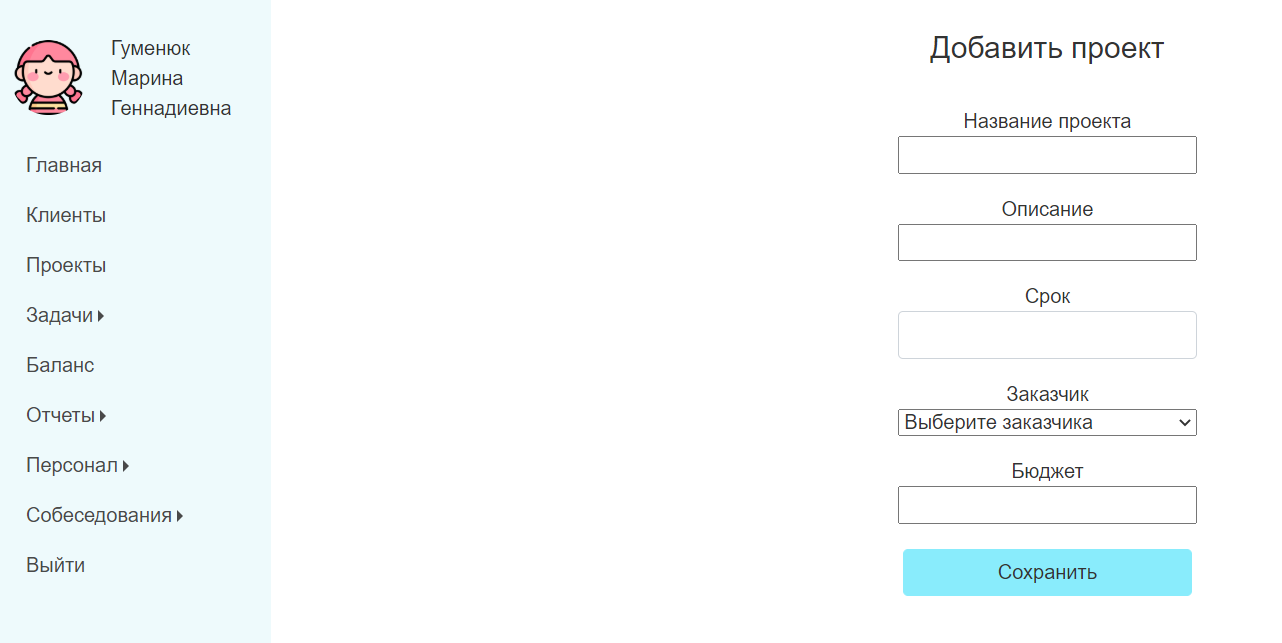


Рисунок 5.3.8 – Створення нового проекту

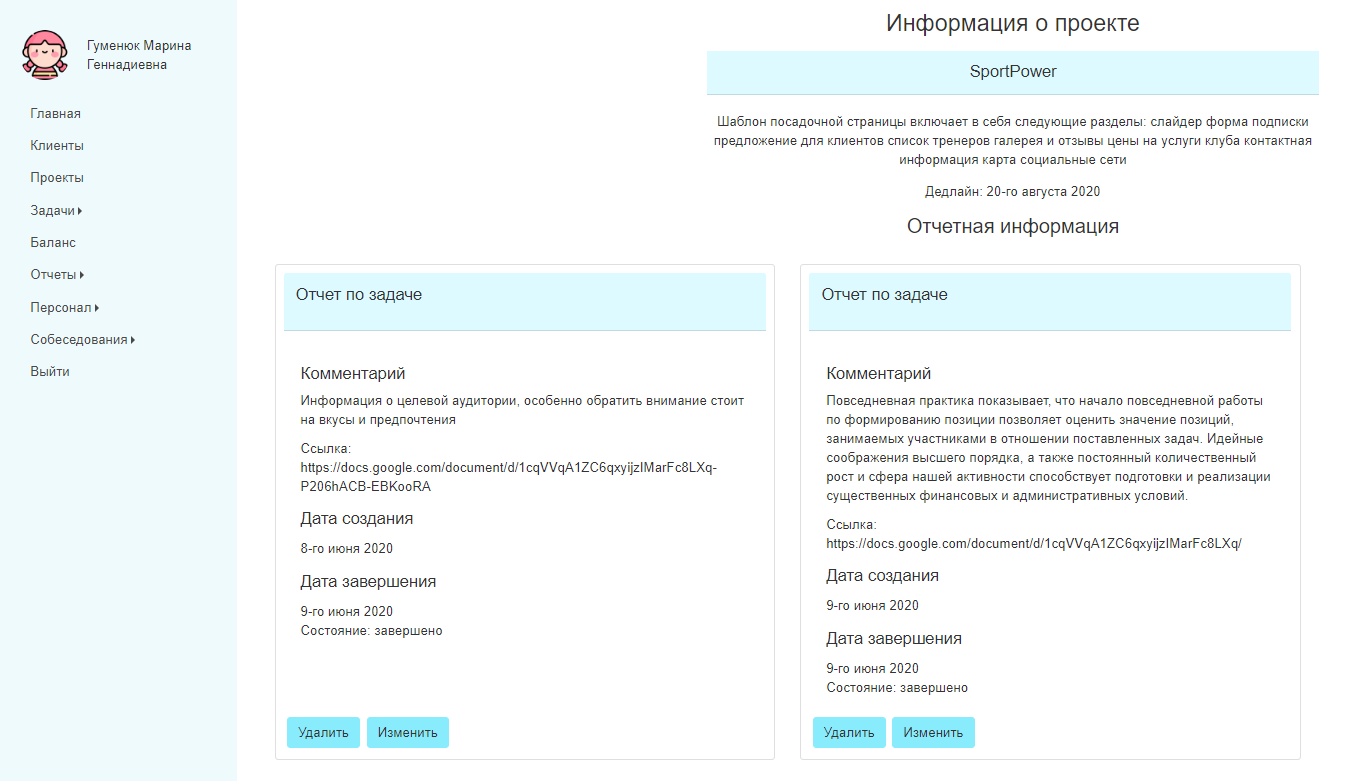


Рисунок 5.3.9 – Перегляд детальної інформації про проект

Кожен виконавець може переглядати назначені йому задачі, що потребують виконання (рисунок 5.3) і задачі, що були виконані (рисунок 5.3). Менеджер може бачити абсолютно всі існуючі задачі:

const tasks = await Tasks.find().lean();

Користувачам доступні лише ті задачі, які були створені спеціально для виконання цими користувачами:

const tasks = await Tasks.find({ employee: req.params.employeeId }).lean();

Даний функціонал обробляється контроллером task.controllers.js.

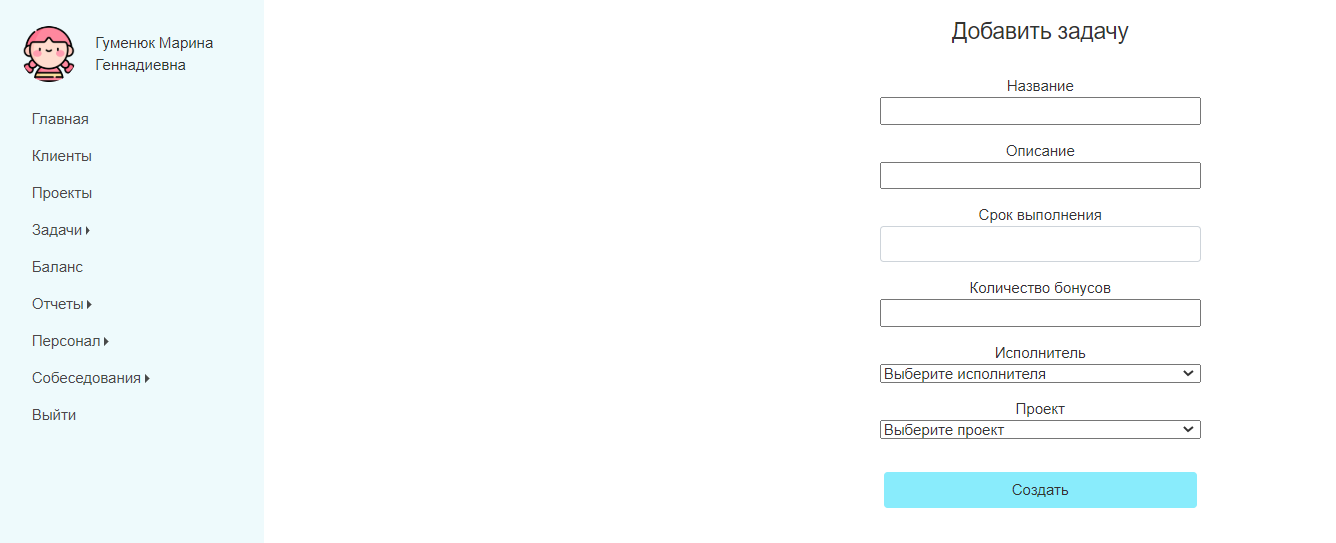


Рисунок 5.3.10 – Створення задачі

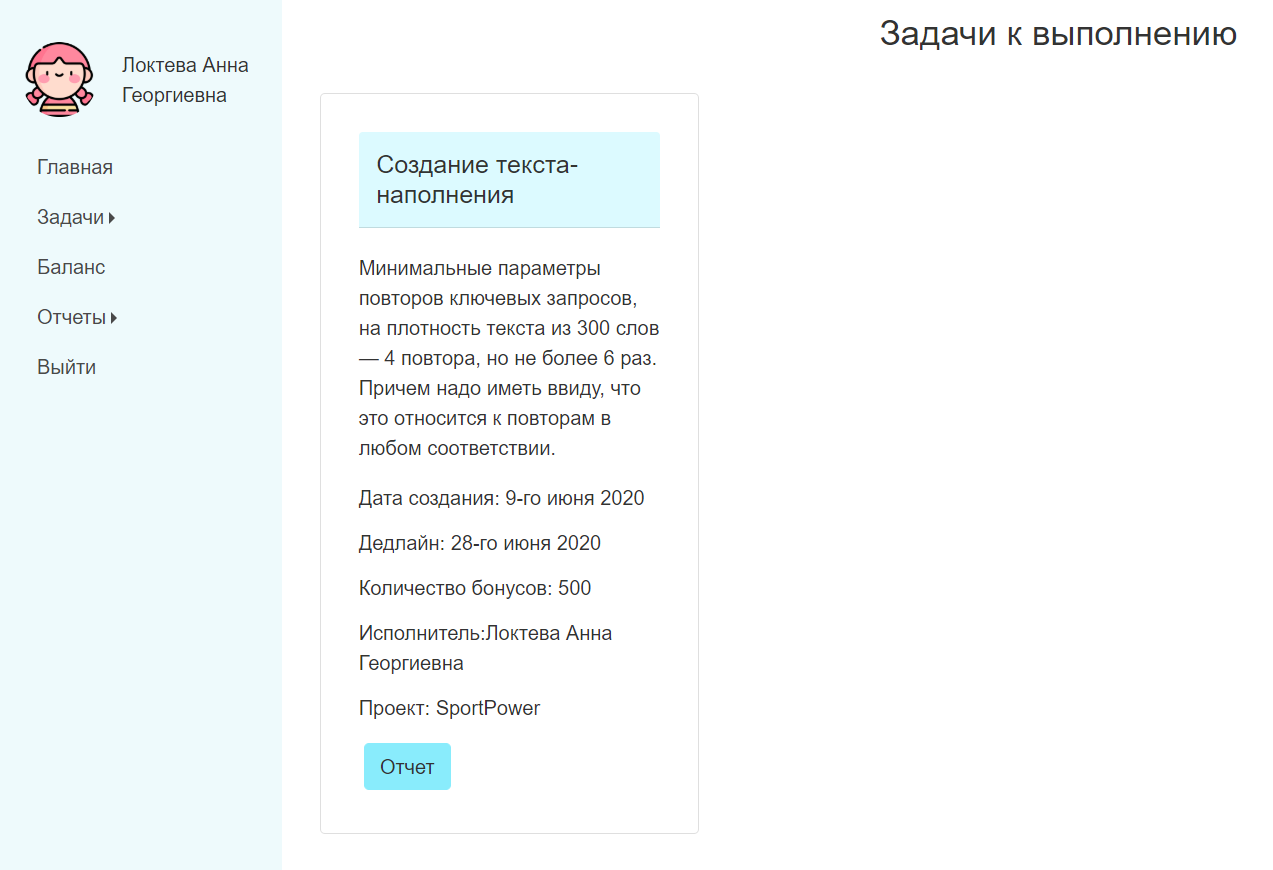


Рисунок 5.3.11 – Задачі для виконання

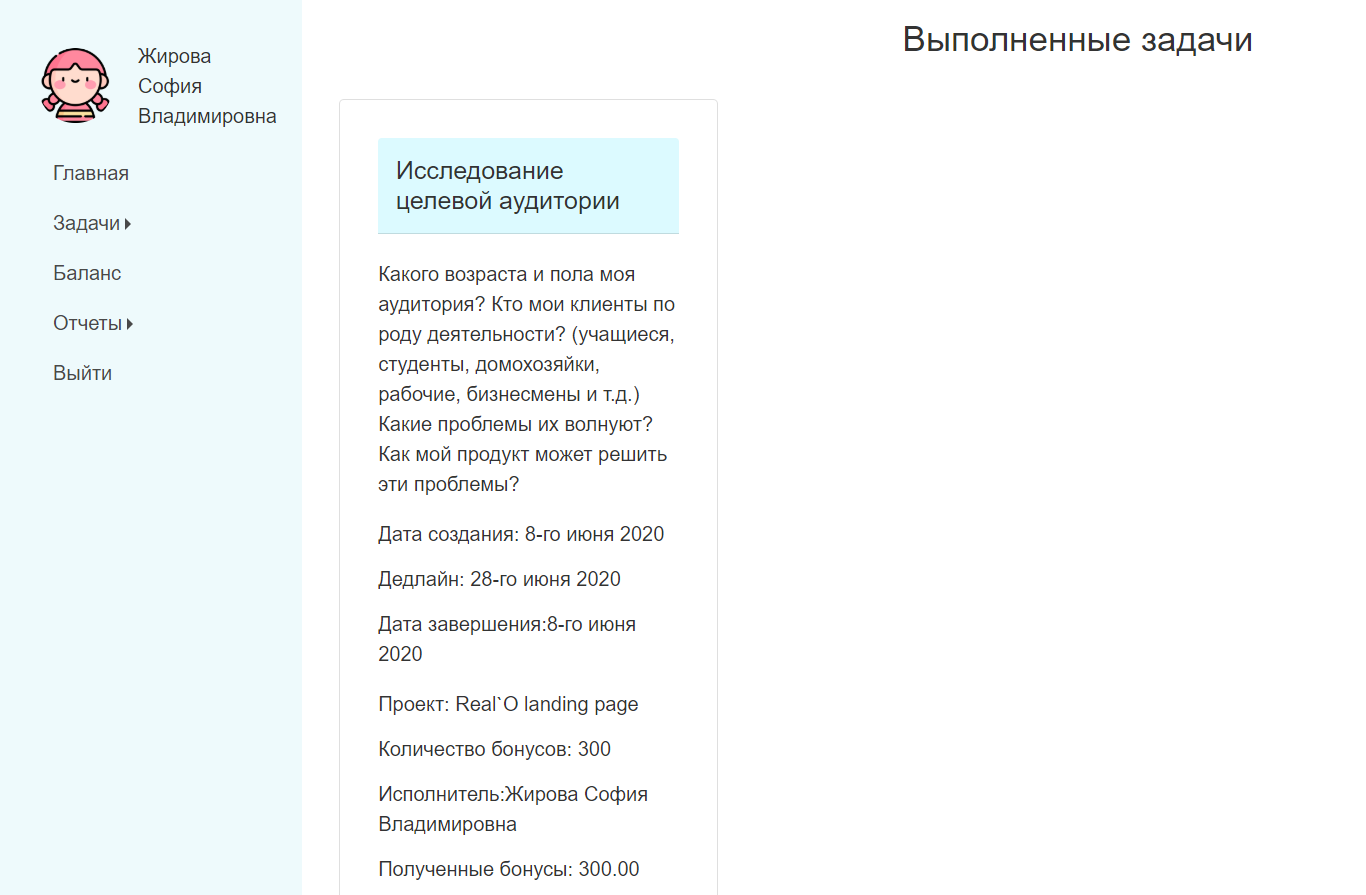


Рисунок 5.3.12 – Виконані задачі

Задача вважається завершеною, коли завершено звіт по ній. Створення, оновлення та завершення звітів забезпечує контроллер reports.controller.js.



Рисунок 5.3.13 – Створення звіту виконання задачі

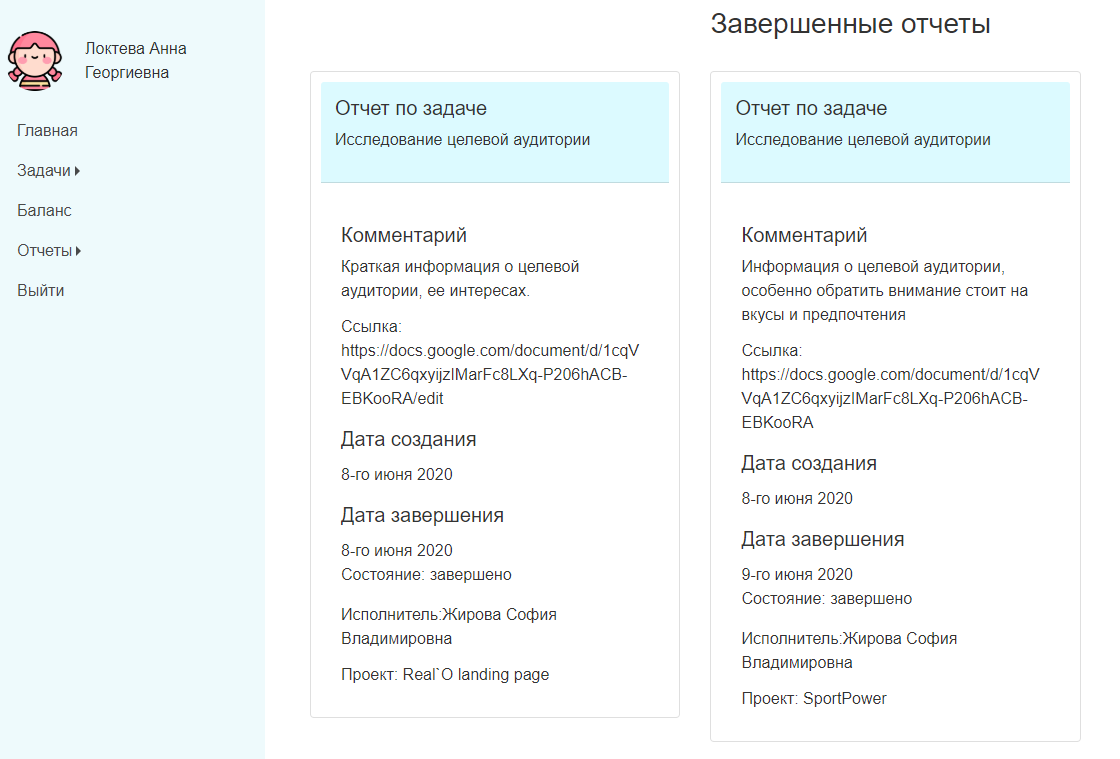


Рисунок 5.3.14 – Перегляд завершених звітів

Кожен користувач системи може подивитись суму бонусів, штрафів, очікувані бонусні виплати та доступні за виконання задач бонуси (рисунок 5.3.15). За даний функціонал відповідає контролер bounce.controller.js. Розрахунок бонусів та штрафів відбувається за наступною реалізацією (формула 1.1):

module.exports.complete = async(id) => {

const task = await get(id);

const plannedBonuces = task.bonuce;

const createDate = moment(task.createdAt);

const deadline = moment(task.deadline);

const currDate = moment();

const realBonuces = deadline.diff(currDate, "day");

const maxBonuces = deadline.diff(createDate, "day");

const earnedBonuce = plannedBonuces + (plannedBonuces / maxBonuces) \* (realBonuces - maxBonuces);

return update(id, {

status: "finished",

finishDate: currDate.toDate(),

earnedBonuce,

});

}

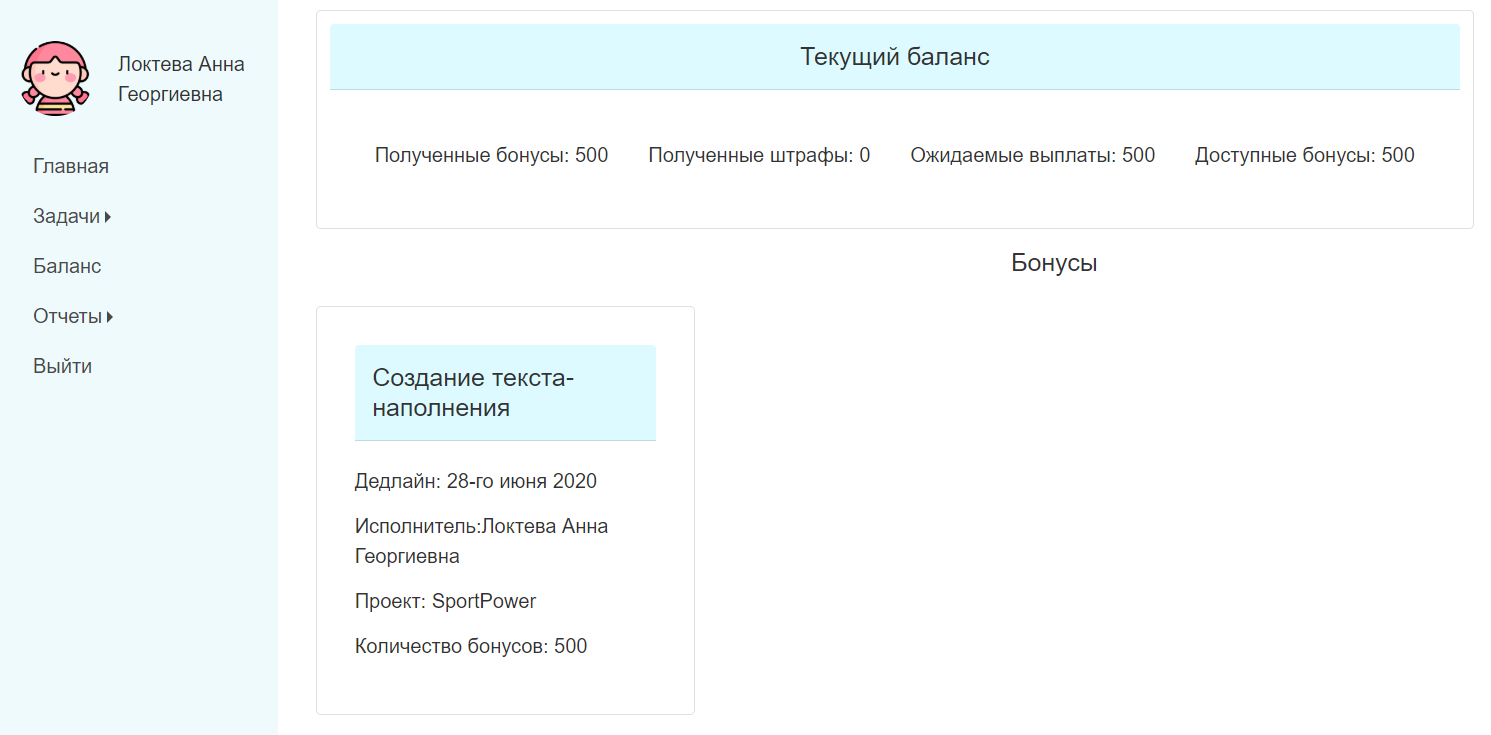


Рисунок 5.3.15 – Перегляд балансу бонусної системи

Система передбачає можливість перегляду видалення та оцінювання співробітників (рисунок 5.3.16). Створення нових співробітників буде розглядатись нижче (при роботі з резюме потенційних кандидатів). Даний функціонал забезпечує контролер employees.controller.js.

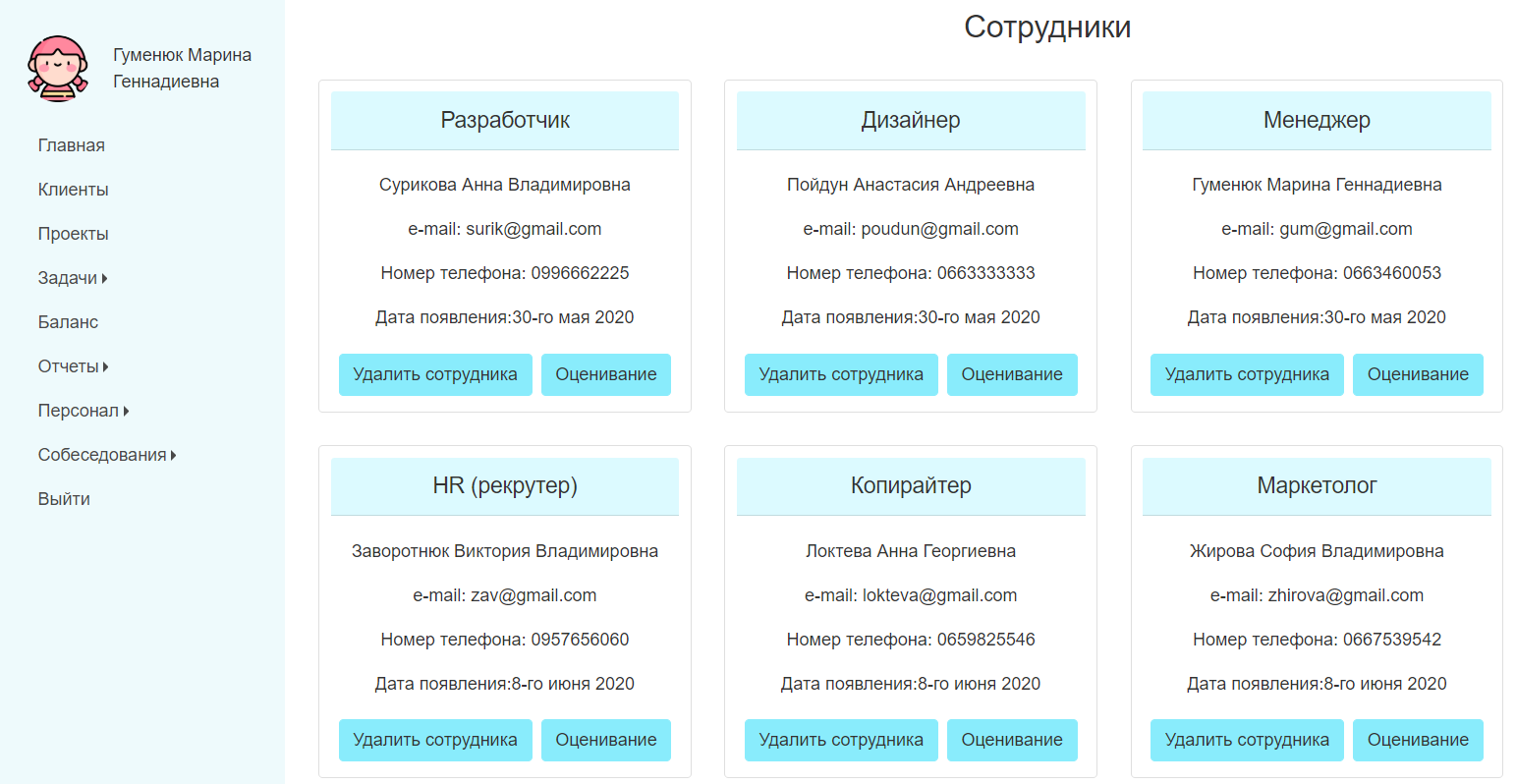


Рисунок 5.3.16 – Перегляд співробітників

За оцінювання співробітників також відповідає контролер criterias.controller.js, який дозволяє створювати та видаляти критерії оцінювання робітників. Також за допомогою нього можна побачити історію оцінювання персоналу (рисунок 5.3.17 – 5.3.19). Відображення історії реалізується наступним чином:

getGradesData() {

const { data } = this.props;

const group = \_.chain(data)

.groupBy((el) => moment(el.createdAt).format("Do MMMM YYYY"))

.map((el, key) => ({

title: key,

data: el,

}))

.value();

return group;

}

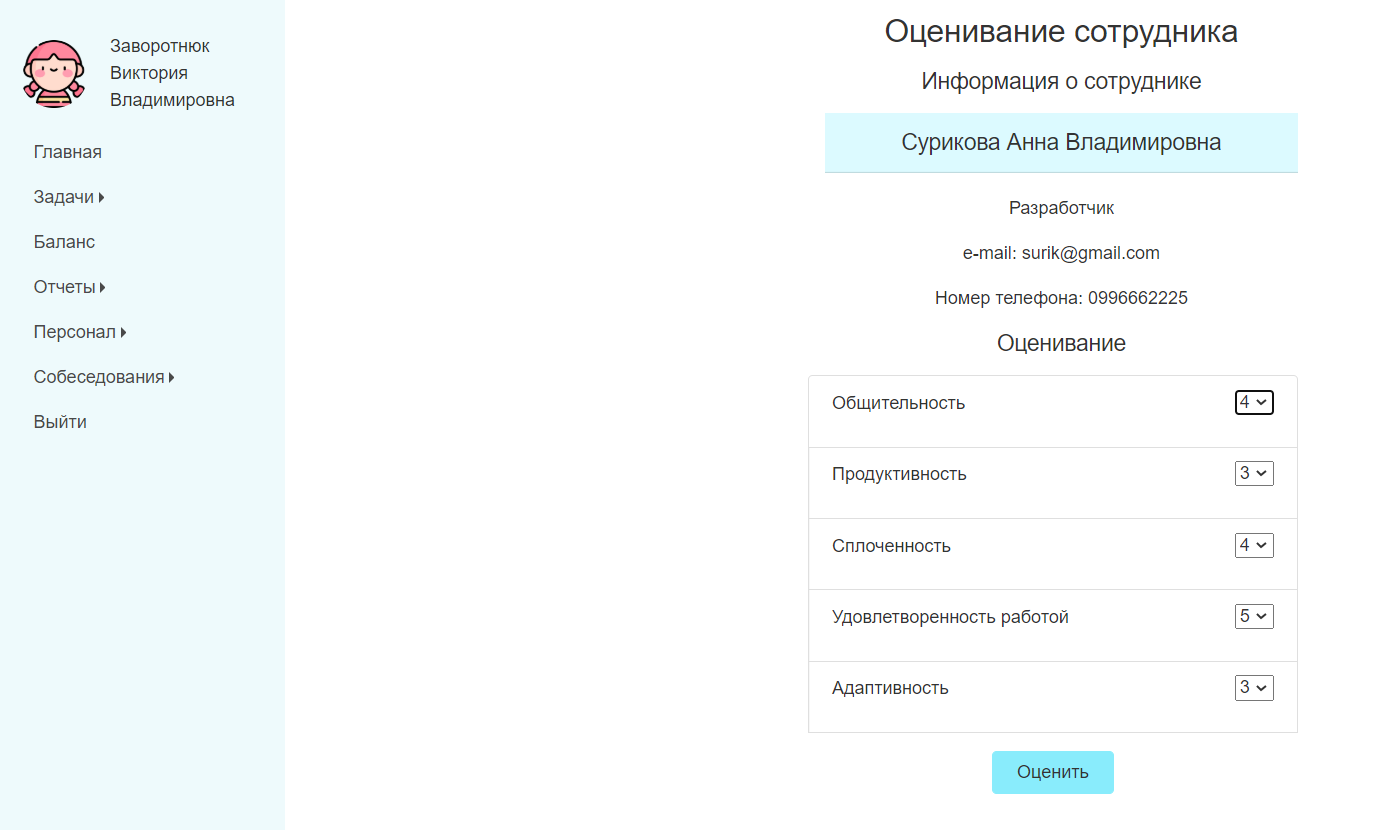


Рисунок 5.3.17 – Оцінювання співробітників

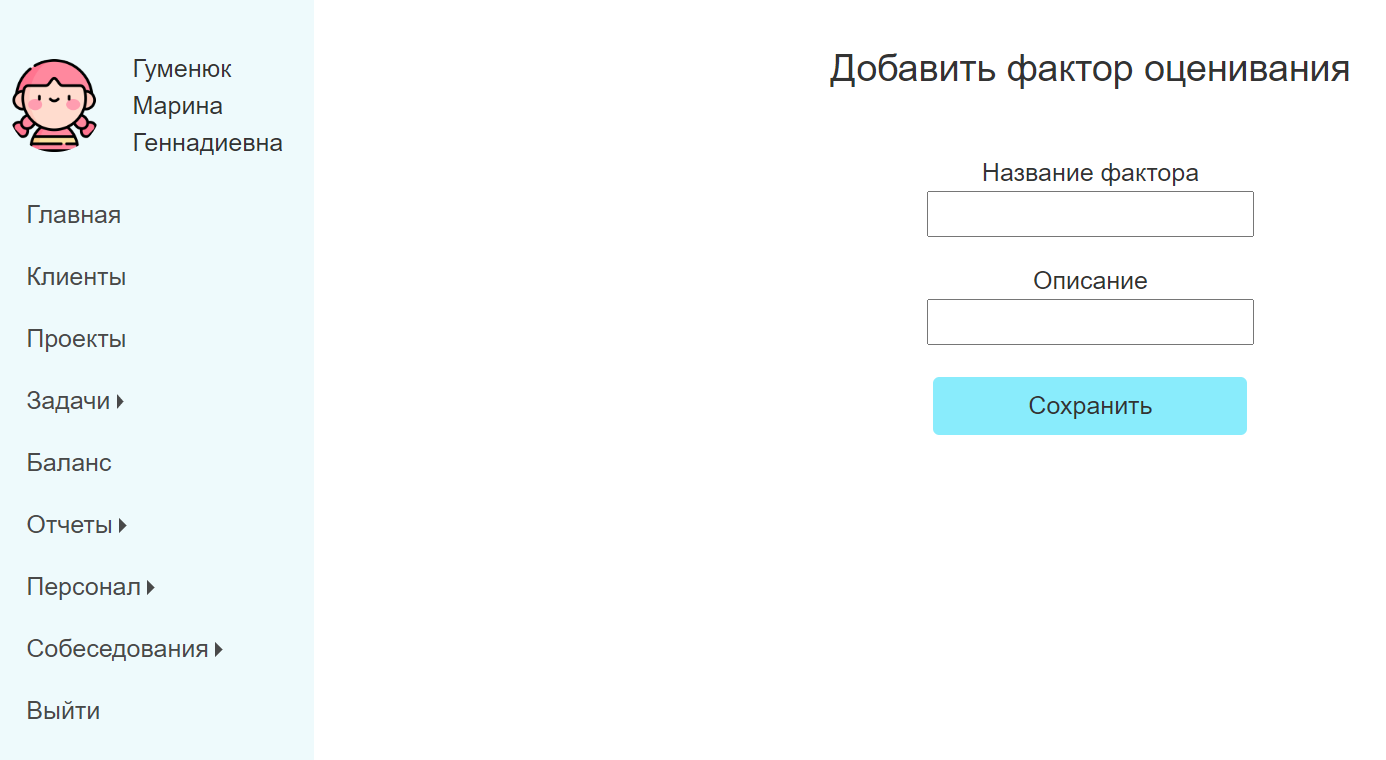


Рисунок 5.3.18 – Створення критерію оцінювання

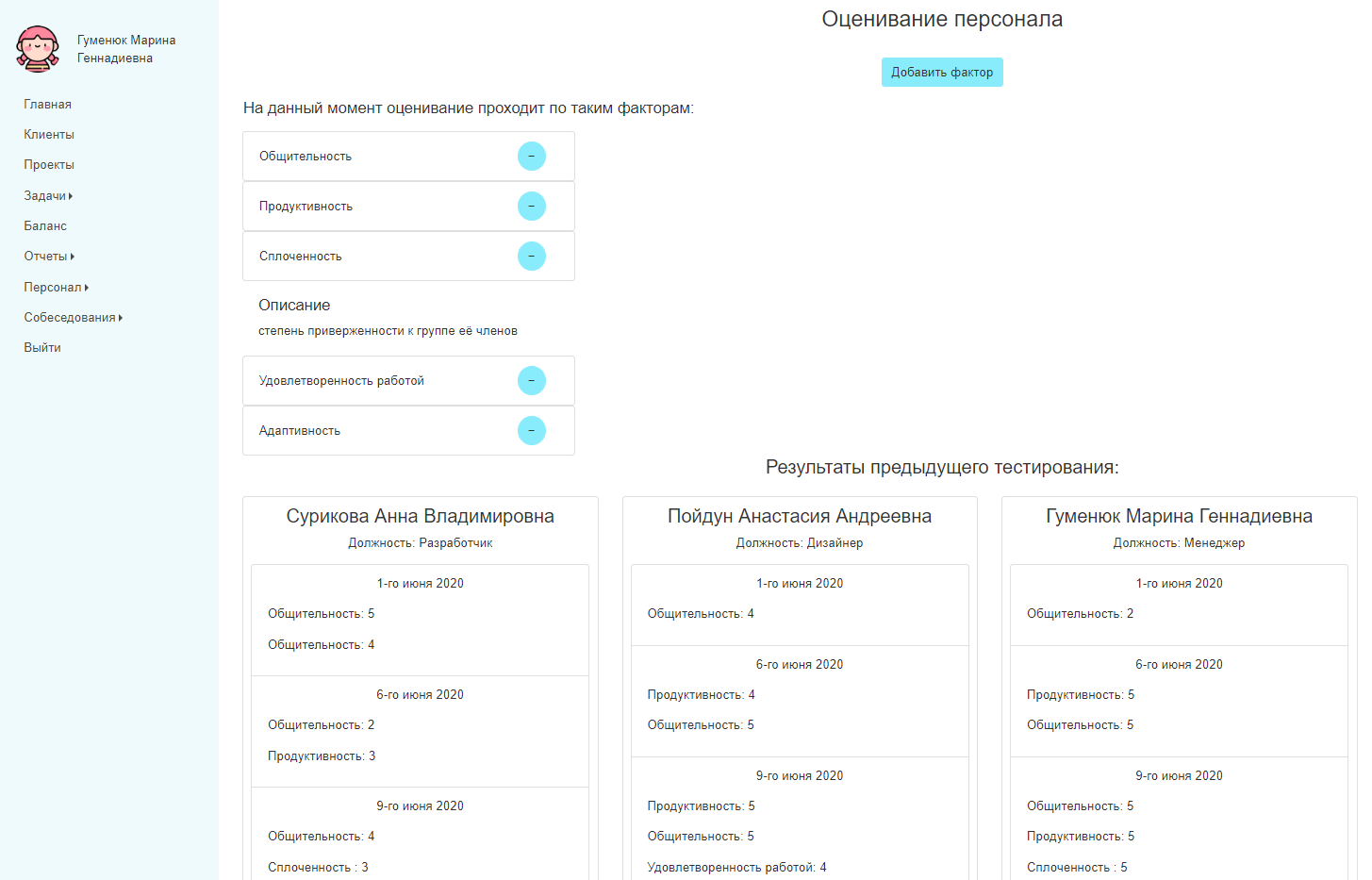


Рисунок 5.3.19 – Перегляд історії оцінювання співробітників

Також система передбачає функцію створення резюме для потенційних кандидатів та перегляд існуючих резюме кандидатів (рисунок 5.3.20-5.3.21). Це реалізується за допомогою контроллера reviews.controller.js.

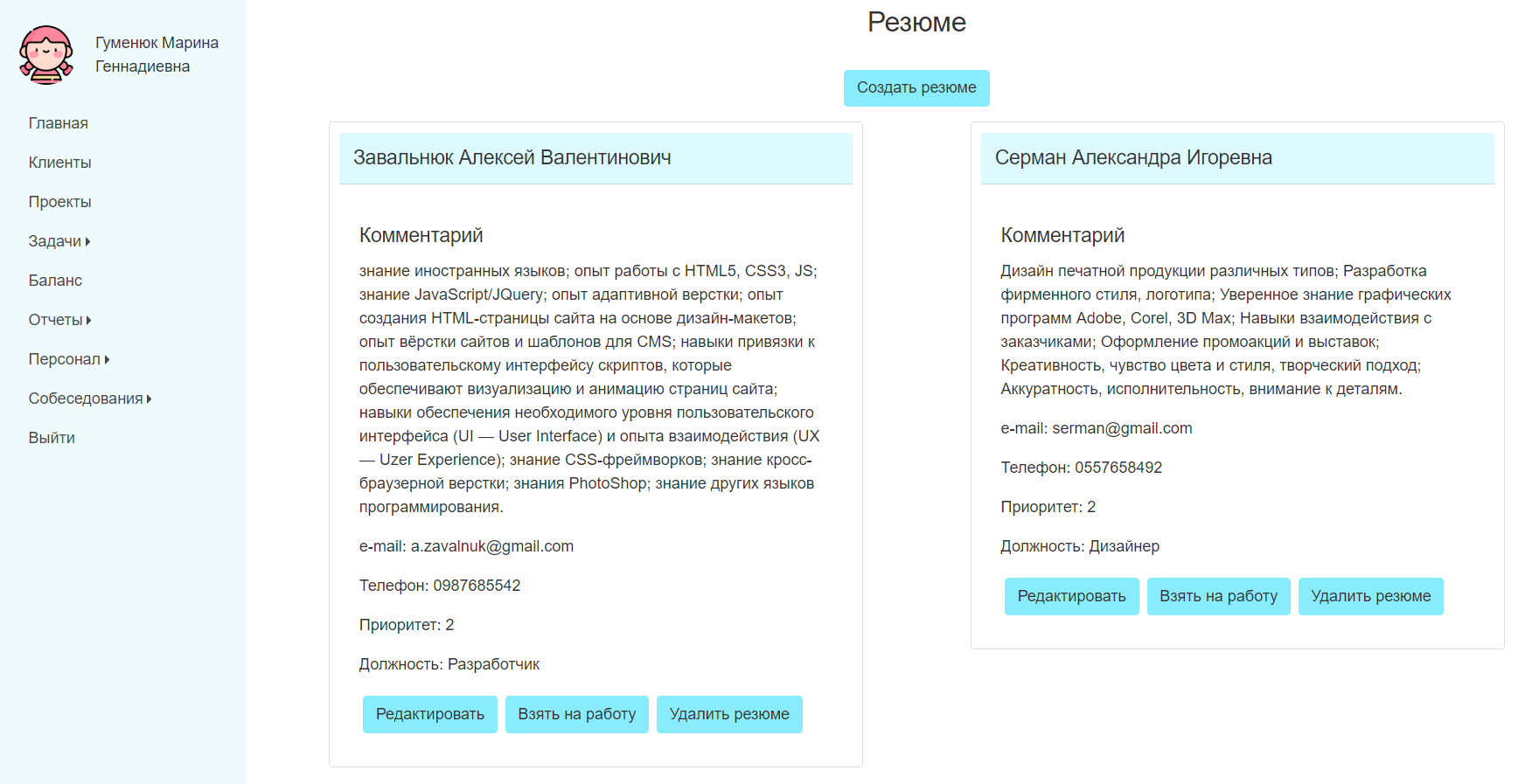


Рисунок 5.3.20 – Перегляд резюме потенційних кандидатів на вакансію

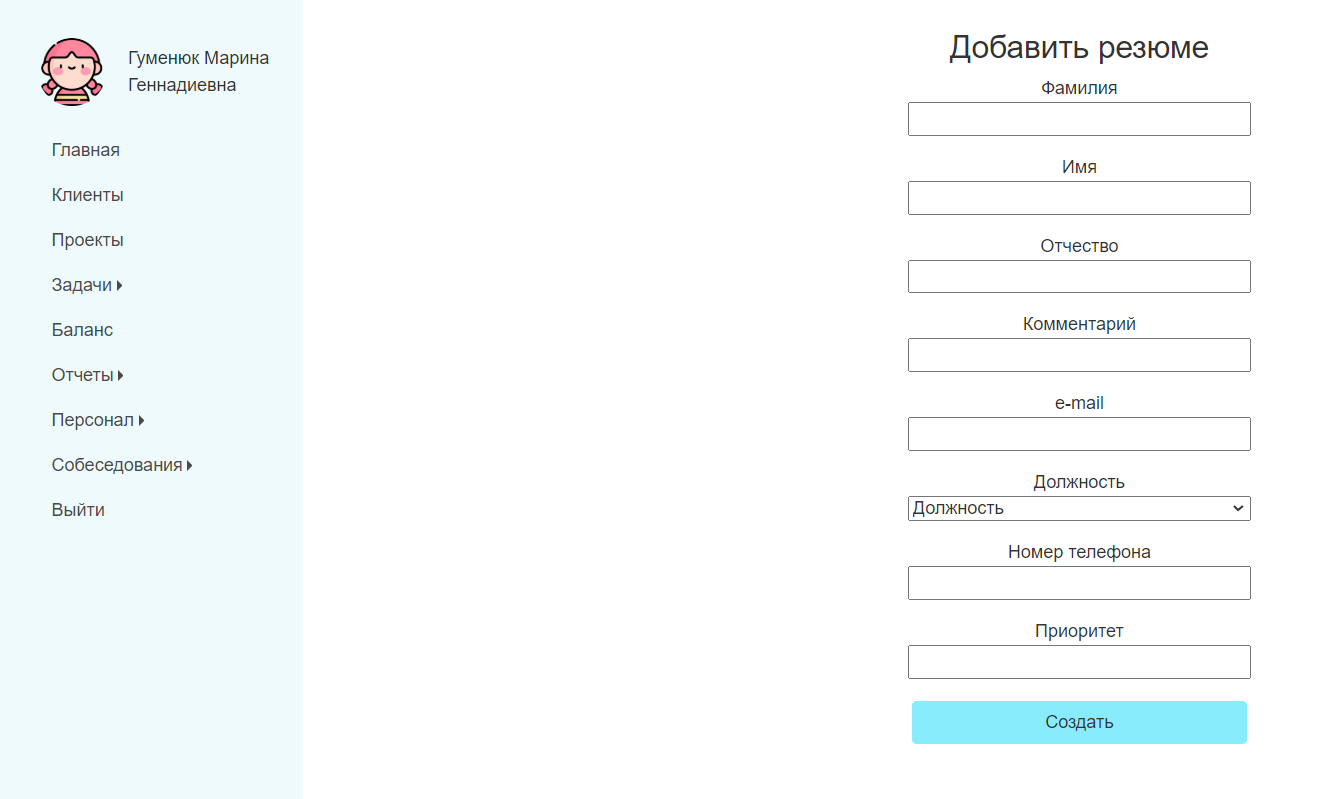


Рисунок 5.3.21 – Створення нового резюме

Система дозволяє планувати, створювати та видаляти події у розкладі співбесід. Кожен день має прев’ю – цифру, що означає кількість запланованих подій на той чи інший день. Натиснувши на нього, можна передивитись детальну інформацію щодо події і по її завершенню, залишити коментар і відзначити результати проведеної співбесіди. Даний функціонал реалізується за допомогою контролеру calendar.controller.js. Результат проведеної співбесіди та коментар (за потребою) реалізується таким чином:

resultFunc = async (id, value) => {

const input = document.getElementById(id);

try {

await CalendarService.update(id, {

result: value,

description: input.value,

});

this.loadInfo();

} catch {}

};

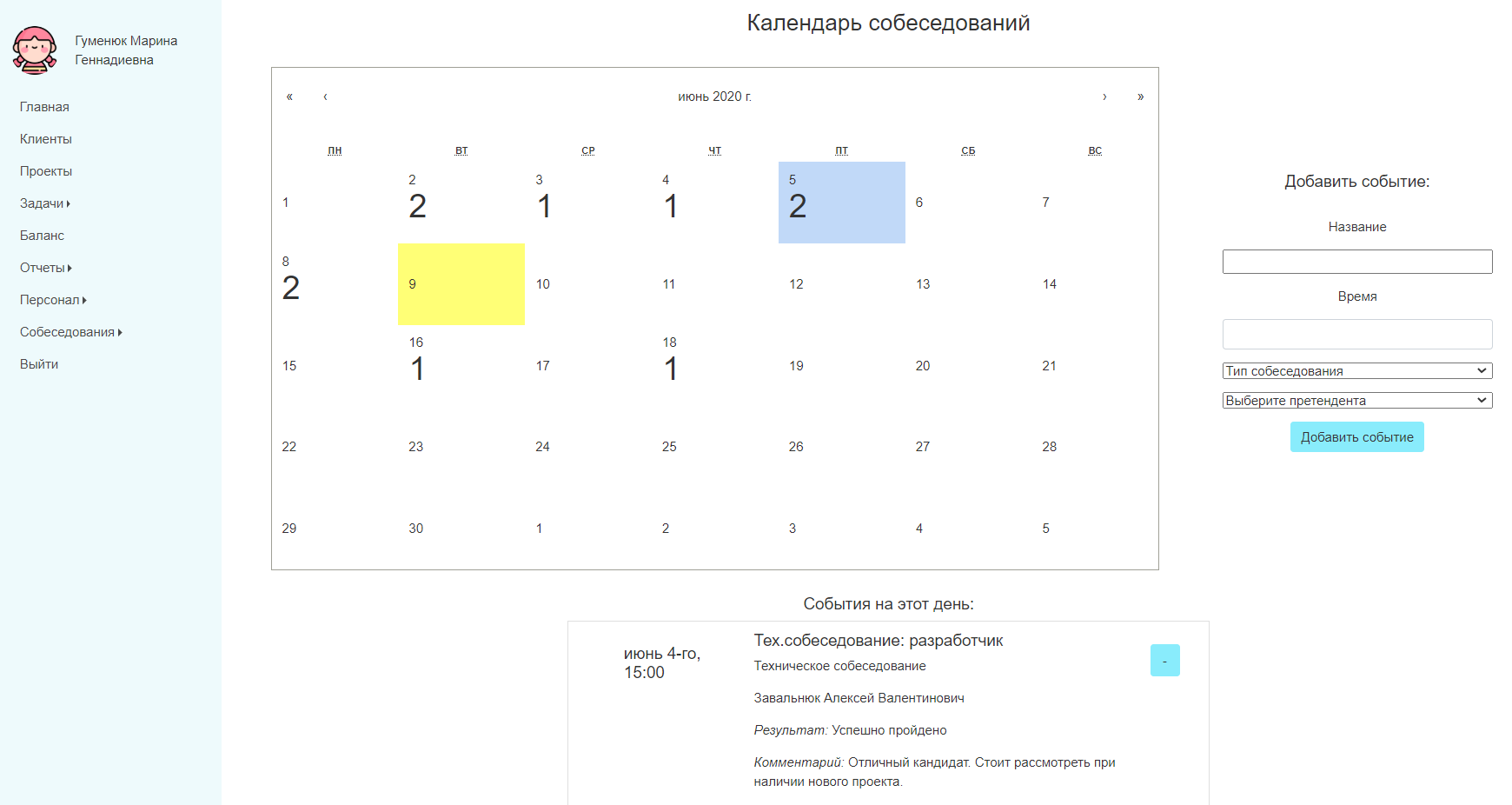


Рисунок 5.3.22 – Перегляд та створення подій

## 5.5 Розгортання інформаційної системи за допомогою хмарних технологій Heroku та Git

CI/CD [[8]](#_That_CI/CD_Thingy:) - концепція, яка реалізується як конвеєр, полегшуючи злиття тільки що закоміченного коду в основну кодову базу. Концепція дозволяє запускати різні типи тестів на кожному етапі (виконання інтеграційного аспекту) і завершувати його запуском з розгортанням закоміченного коду в фактичний продукт, який бачать кінцеві користувачі (виконання доставки).

CI/CD необхідні для розробки програмного забезпечення із застосуванням Agile-методології, яка рекомендує використовувати автоматичне тестування для швидкого налагодження робочого програмного забезпечення. Автоматичне тестування дає зацікавленим особам доступ до новоствореним функцій і забезпечує швидку зворотний зв'язок.

CI/CD має в основі 3 принципи:

1) своєчасна участь різних зацікавлених сторін в будь-якому проекті - кожен етап конвеєрної обробки CI/CD створює середовище, налаштоване на те, щоб групи брали відповідальність за відповідну стадію тестування, забезпечуючи цілісність процесу;

2) зниження ризику - створення логічних та письмових тестів;

3) короткий цикл зворотного зв'язку – необхідність автоматизувати все, що можна автоматизувати; за допомогою масштабування скорочується час, що витрачається на тестування програмного забезпечення, що дозволяє автоматизувати процес розгортання.

Heroku - це хмарна платформа, яка дозволяє компаніям створювати, розгортати, контролювати та масштабувати їхні продукти, це платформа для даних, а також додатків, що забезпечує безпечну масштабовану базу даних як послугу з багатьма інструментами розробників.

Коли розробники щось створюють (наприклад, додаток), вони вносять постійні зміни в код, випускаючи нові версії до і після першого офіційного (не бета-версії) випуску.

Системи управління версіями проводять ці редакції зберігаючи модифікації в центральному сховищі. Це дозволяє розробникам легко співпрацювати, оскільки вони можуть завантажити нову версію програмного забезпечення, внести зміни та завантажити останню версію. Кожен розробник може побачити ці нові зміни, завантажити їх та зробити свій внесок. GitHub дає можливість зберігати зміни та код на віддаленному репозиторії.

Для того, щоб розгорнути дану інформаційну систему, необхідно створити та завантажити код на віддаленний git-репозиторій. Безпосередньо на Heroku необхідно створити новий проект та під’єднати його до GitHub репозиторія, де знаходиться додаток. Для того щоб вказати сервісу Heroku, за допомогою якої команди відбувається початковий запуск аплікації, створюється Procfile, в якому вказується команда запуску.

web: npm start

Після цього в налаштуваннях проекту необхідно вказати авторозгортання при посиланні нових змін на основну гілку проекту на github. (Рис. 5.3.1) При додаванні змін до master-гілки у GitHub репозиторії Heroku завантажує собі останні зміни, визначає проект як node.js аплікацію, виконує інсталювання усіх необхідних пакетів, описаних у package.json, запускає команду npm run build для генерації статичних файлів на основі клієнтської частини аплікації. Після цього запускається команда запуску, вказана у Procfile. При створенні користувачем запиту до серверу, у разі, коли даний запит не обробляється ні одним із наявних контролерів, клієнт отримує згенеровані статичні файли клієнтської частини аплікації.

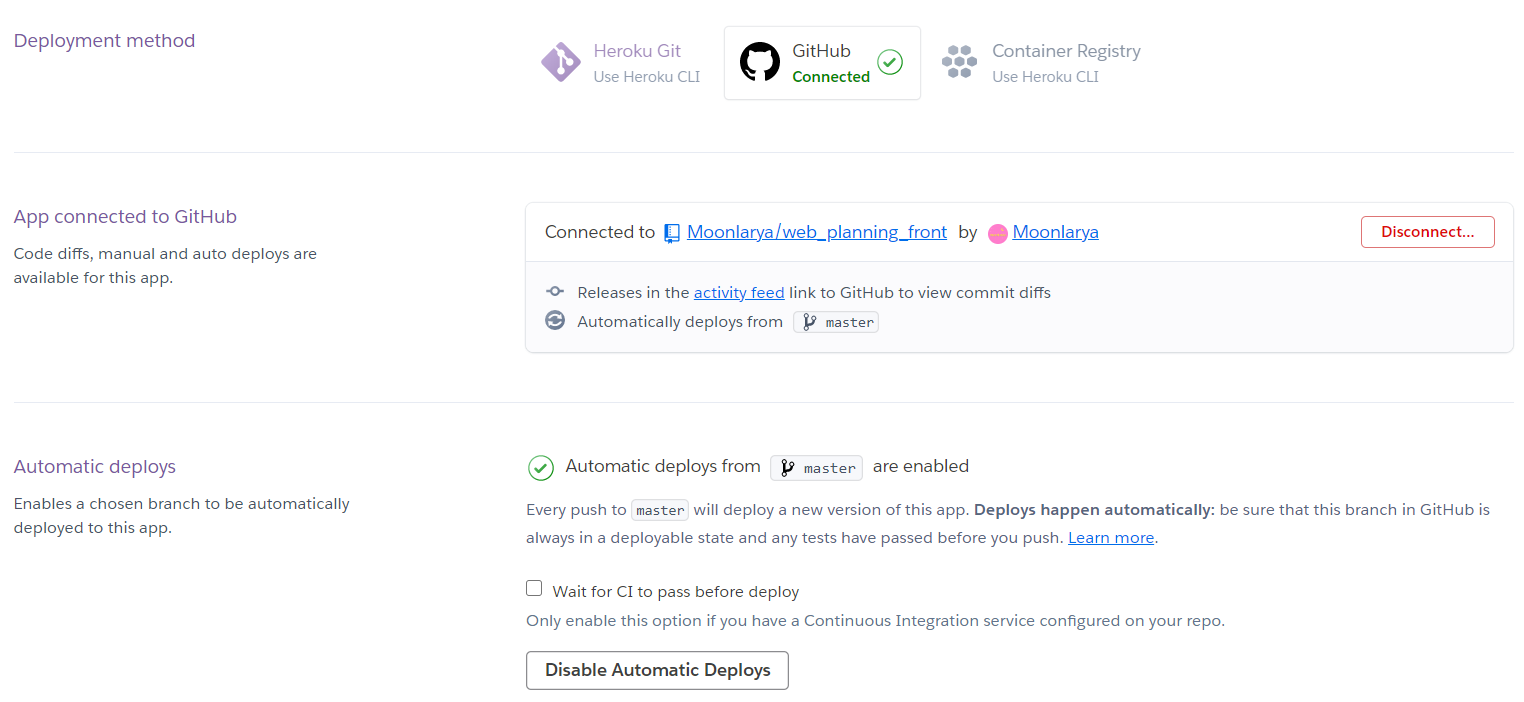


Рисунок 5.5.1 – Налаштування в Heroku Dashboard

# 6 РЕАЛІЗАЦІЯ БЕЗПЕКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

### 6.1 Алгоритм шифрування Bcrypt для захисту інформації

Для забезпечення захищеності паролі не повинні зберігатися в базі даних у відкритому вигляді. Для запобігання цього, використовується алгоритм bcrypt – це адаптивна криптографічна хеш-функція формування ключа, що використовується для захищеного зберігання паролів, основана на шифрі Blowfish.

Для криптографічних хеш-функцій є три додаткових умови, які відрізняють їх від всіх інших:

1) незворотність: для заданого значення хеш-функції m повинно бути обчислювально нездійсненно знайти блок даних X, для якого H (X) = m;

2) стійкість до колізій першого роду: для заданого повідомлення M має бути обчислювально нездійсненно підібрати інше повідомлення N, для якого H (N) = H (M);

3) стійкість до колізій другого роду: має бути обчислювально нездійсненно підібрати пару повідомлень ~ (M, M '), що мають однаковий хеш.

Для реалізації шифрування за допомогою bcrypt використовується бібліотека node-bcrypt-js. Вона створює хеш на основі паролю користувача, який дозволить порівняти його з паролем, що був введений при авторизації та зробити висновок про коректність введених даних. Ця бібліотека підтримує $ 2a $ і $ 2b $ префікси хешів bcrypt. Сумісність з хешами, створеними іншими мовами, не гарантується на 100% через різницю в кодуванні символів.

## 6.2 SSL

Для безпеки запитів використовується протокол HTTPS - прикладний протокол передачі даних в мережі, що підтримує шифрування за допомогою криптографічних протоколів SSL і TLS.

Забезпечення безпечної передачі даних необхідно на сайтах, де вводиться і передається конфіденційна інформація (особисті дані користувачів, деталі доступу, реквізити платіжних карт) - на будь-яких сайтах з авторизацією, взаємодією з платіжними системами, поштовими сервісами. Шифрування таких даних дозволить запобігти їх отримання і використання третіми особами.

Для реалізації передачі даних за допомогою HTTPS на веб-сервері, що обробляє запити від клієнтів, повинен бути встановлений спеціальний SSL-сертифікат.

SSL - криптографічний протокол, який забезпечує більш безпечний зв'язок. Він використовує асиметричну криптографію для аутентифікації ключів обміну, симетричне шифрування для збереження конфіденційності, коди аутентифікації повідомлень для цілісності повідомлень.

Вся конфіденційна інформація, така як ключі шифрування та посилання на сервер бази даних знаходяться у файлі змінних середовища .env:

JWT\_SECRET=PASSWORD123

BCRYPT\_SALT\_WORK\_FACTOR = 10

MONGODB\_URL=mongodb+srv://manager:mymongobd.mongodb.net/test

# ВИСНОВКИ

У результаті виконання дипломної роботи створена система для планування та контролю задач на мікропідприємстві.

Для створення системи використані наступні засоби:

1. архітектура REST;
2. стек технологій MERN (MongoDB, Express JS, React JS, Node.js);
3. шаблон проектування MVC;
4. хмарна PaaS-технологія Heroku;

Система планування і контролю виконання завдань для мікропідприємства (веб-студії) через веб-інтерфейс надає наступні інструменти:

* для менеджера – розподіл завдань між співробітниками, контроль їх виконання, моніторинг процесу роботи над проектами;
* для виконавця – облік поставлених завдань, надання звітності про результати роботи, відстеження власного прогресу і доступних матеріальних заохочень;
* для рекрутера – ведення календаря співбесід, історії результатів співбесід, журналу періодичного оцінювання особистих і професійних якостей співробітників.

Створений програмний продукт призначений для використання в якості інструменту для автоматизації робочих процесів в веб-студії і може бути адаптований з урахуванням специфіки мікропідприємств різних видів інтернет-діяльності.

Результати розробки програмного продукту були обговорені на сімнадцятій всеукраїнській конференції студентів і молодих науковців, тези доповіді опубліковані [11].

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

###### Customer Relationship Management (CRM) Software Configuration [Електронний ресурс] / Режим доступу: <https://www.trustradius.com/crm - 29.05.2020>

###### 40 сервисов для управления задачами и проектами [Електронний ресурс] / Режим доступу: [https://vc.ru/services/50333-40-servisov-dlya-upravleniya-zadachami-i-proektami - 01.06.2020](https://vc.ru/services/50333-40-servisov-dlya-upravleniya-zadachami-i-proektami%20-%2001.06.2020)

###### Микропредприятия, малые, средние и крупные предприятия в новой классификации предприятий, принятой с изменениями в законе о бухгалтерском учете [Електронний ресурс] / Режим доступу: <https://byhgalter.com/ru/mikropredpriyatiya-malye-srednie-i-krupnye-predpriyatiya-v-novoj-klassifikacii-predpriyatij-prinyatoj-s-izmeneniyami-v-zakone-o-buxgalterskom-uchete/> - 29.05.2020

###### BPM Tutorials with ARIS [Електронний ресурс] / Режим доступу: <https://www.ariscommunity.com/university/tutorial - 18.05.2020>

###### ECMAScript 6 - ECMAScript 2015[Електронний ресурс] / Режим доступу: [https://www.w3schools.com/js/js\_es6.asp - 19.05.2020](https://www.w3schools.com/js/js_es6.asp%20-%2019.05.2020)

###### Как работает Virtual DOM? [Електронний ресурс] / Режим доступу: <https://medium.com/@abraztsov/how-virtual-dom-work-567128ed77e9> - 20.05.2020

###### Client-Server Web Apps with JavaScript and Java: Rich, Scalable, and RESTfulISBN-10: 9781449369330 ISBN-13: 978-1449369330 [Текст] / O`Relly media – Sebastopol, Kalifornia: 2014. – 260 с.

###### That CI/CD Thingy: Principles, Implementation & Tools [Електронний ресурс] / Режим доступу: [https://blog.gds-gov.tech/that-ci-cd-thing-principles-implementation-tools-aa8e77f9a350 - 22.05.2020](https://blog.gds-gov.tech/that-ci-cd-thing-principles-implementation-tools-aa8e77f9a350%20-%2022.05.2020)

###### What is MERN stack? [Електронний ресурс] / Режим доступу: <https://medium.com/@digimktg/what-is-mern-stack-9c867dbad302> - 23.05.2020

###### Deploying React with Zero Configuration [Електронний ресурс] / Режим доступу: [https://blog.heroku.com/deploying-react-with-zero-configuration – 20.05.2020](https://blog.heroku.com/deploying-react-with-zero-configuration%20–%2020.05.2020)

###### What is JSON Web Token? [Електронний ресурс] / Режим доступу: <https://jwt.io/introduction/> - 25.05.2020

###### SASS/SCSS Documentation - [Електронний ресурс] / Режим доступу: <https://sass-lang.com/guide> - 20.05.2020

###### Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript. [Текст] / СПб.: Питер, 2017. — 336 с.

###### Информационные системы и базы данных: организация и проектирование: учеб. пособие. [Текст] / СПб.: БХВ-Петербург, 2009. — 528 с.

###### Node.js Docs [Електронний ресурс] / Режим доступу: <https://nodejs.org/en/docs/> - 17.02.2020

###### React.js Документация [Електронний ресурс] / Режим доступу: [https://ru.reactjs.org/docs/getting-started.html - 24.03.2020](https://ru.reactjs.org/docs/getting-started.html%20-%2024.03.2020)

###### Справочник API [Електронний ресурс] / Режим доступу: [https://expressjs.com/ru/api.html - 28.02.2020](https://expressjs.com/ru/api.html%20-%2028.02.2020)

###### Documentation JS [Електронний ресурс] / Режим доступу: <https://documentation.js.org/> - 03.05.2020

###### Гуменюк М. Г, Розновец О.И. Система планирования и контроля выполнения задач для микропредприятия/ Інформатика, інформаційні системи та технології: тези доповідей сімнадцятої всеукраїнської конференції студентів і молодих науковців. Одеса, 24 квітня 2020 р. – Одеса, 2020. – с. 180-181

# ДОДАТОК А

Історії користувачів

Я как копирайтер/дизайнер/…/маркетолог (любой участник команды, кроме менеджера):

1) Хочу иметь инструмент для отслеживания поставленных задач менеджером и их сроков выполнения с целью составления индивидуального рабочего графика.

Входные данные: ФИО сотрудника.

Выходные данные: Задача (ФИО сотрудника, задача, срок выполнения)

2) Хочу иметь инструмент для отслеживания собственной продуктивности (на сколько я успеваю по срокам выполнения, сколько было проделано работы) для дальнейшей корректировки плана загруженности.

Входные данные: ФИО сотрудника

Выходные данные: Задача (ФИО сотрудника, задача, статус (начато/в процессе/завершено), срок выполнения, сколько осталось до завершения срока/сколько времени сэкономлено), поощрение;

3) Хочу иметь поощрения за проделанную работу с целью получения дополнительной мотивации работать.

Входные данные: задача;

Выходные данные: Задача (ФИО сотрудника, задача, сколько времени сэкономлено), поощрение;

4) Хочу знать информацию о доступных поощрениях, с целью узнать как их заполучить.

Входные данные: ФИО сотрудника;

Выходные данные: Поощрение (Название, описание, что необходимо сделать, сроки выполнения);

5) Хочу иметь возможность отслеживать полученные поощрения с целью мотивации себя к работе.

Входные данные: ФИО сотрудника;

Выходные данные: Поощрение (Название, описание, что сделано, сроки выполнения);

6) Хочу иметь возможность создания, редактирования, удаления данных в соответствии с моими полномочиями с целью актуализации информации о рабочем процессе.

Входные данные: ФИО сотрудника;

Выходные данные: записи сотрудника;

Я как копирайтер:

1) Хочу получать информацию от маркетологов о целевой аудитории, социальных опросах и конкурентах на рынке, чтоб создать максимально уникальный продукт (текст).

Входные данные: Задача (ФИО сотрудника, задача)

Выходные данные: Информация (Информация от маркетолога с комментариями в виде ссылки на внешний документ)

2) Хочу иметь инструмент для сохранения своей текущей работы и отправки ее для дальнейшего использования дизайнеру с целью хранения информации на удаленном носителе (чтоб избежать потерь работы) и избегания договоренности передачи данных (чтоб не договариваться с дизайнером лично когда и куда удобно загрузить проделанную работу).

Входные данные: Задача (ФИО отправителя, ФИО получателя, задача, ссылка на внешний документ с информацией от маркетолога);

Выходные данные: нет;

Я как дизайнер:

1) Хочу получать информацию от маркетологов о целевой аудитории, социальных опросах и конкурентах на рынке, и тексты для сайтов от копирайтеров чтоб создать максимально уникальный продукт (дизайн) для увеличения посещаемости сайта.

Входные данные: Задача (ФИО сотрудника, задача), Информация (Информация от маркетолога с комментариями в виде ссылки на внешний документ, информация от копирайтера в виде ссылки на внешний документ);

Выходные данные: Задача (ФИО сотрудника, задача, ссылка на внешний документ с макетом);

Я как web-разработчик:

1) Хочу получать дизайн для сайтов от копирайтеров, прикрепленных к определенным проектам с целью развития этого проекта (то есть верстки).

Входные данные: Задача (ФИО отправителя, ФИО получателя, задача, файл с макетом в виде ссылки на внешний документ)

Выходные данные: Задача (ФИО сотрудника, задача, ссылка на репозиторий github с реализацией сайта)

Я как HR:

1) Хочу иметь возможность оперирования резюме по приоритетности с целью отсеивания и планирования проведения собеседований.

Входные данные: Резюме (ФИО кандидата, e-mail, номер телефона, приоритет (высокий/низкий/средний));

Выходные данные: ФИО, дата, вид собеседования;

2) Хочу иметь инструмент для проведения анализа состояния сотрудников (их комфорта, отношений друг с другом, продуктивности) для оптимизации полученных результатов

Входные данные: ФИО сотрудника, критерии оценивания (от 0 до 5), Task (ФИО сотрудника, задача, статус (начато/в процессе/завершено), срок выполнения, сколько осталось до завершения срока/сколько времени сэкономлено);

Выходные данные: Состояние (ФИО сотрудника, ФИО анализируемого, оценка по каждому из критериев (от 1 до 5));

3) Хочу иметь возможность распределения приоритетов резюме для составления графика собеседований.

Входные данные: Резюме (ФИО кандидата, e-mail, номер телефона, приоритет (высокий/низкий/средний));

Выходные данные: ФИО, вид собеседования;

Я как маркетолог:

1) Хочу иметь инструмент для фиксирования собранной информации по целевой аудитории, их проблемами и предпочтениям, конкурентам на рынке с целью актуальности разрабатываемого проекта.

Входные данные: Задача (ФИО сотрудника, задача, срок выполнения)

Выходные данные: Информация (Информация от маркетолога с комментариями в виде ссылки на внешний документ);

Я как менеджер проекта:

1) Хочу иметь инструмент-систему вознаграждений для создания мотивации сотрудников.

Входные данные: Задача (ФИО сотрудника, задача, сколько времени сэкономлено);

Выходные данные: ФИО сотрудника, поощрение (Название, описание, что сделано, сроки выполнения);

2) Хочу иметь возможность распределения задач на членов команды с целью планирования проекта.

Входные данные: информация о сотруднике, информация о задаче;

Выходные данные: Задача (ФИО сотрудника, задача, срок выполнения);

# Додаток Б

Повний код інформаційної системи