

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій і систем

Кафедра програмного забезпечення автоматизованих систем

Пояснювальна записка

до кваліфікаційної випускної бакалаврської роботи

бакалавр з напрямку підготовки 6.050103 «Програмна інженерія»
(освітньо-кваліфікаційний рівень)

на тему: Система тестування рівня кваліфікації працівника: Back-end
WEB-додатку

Виконав: студент 4 курсу, групи ПЗ-154

Напряму підготовки
6.050103 «Програмна інженерія»
(шифр і назва напрямку підготовки)

Студент Линник В.Ю.
(прізвище та ініціали)

Керівник Плаасова Ж.М.
(прізвище та ініціали)

Рецензент Копиця П.О.
(прізвище та ініціали)

Черкаси 2019 року

ЗМІСТ

СПИСОК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ	4
ВСТУП	5
1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ	6
1.1 Характеристика предметної області.....	6
1.2 Аналіз існуючих аналогів	6
1.3 Обґрунтування обраних предметних рішень	13
2 ПОСТАНОВКА ЗАДАЧІ	30
2.1 Загальна характеристика системи.....	30
2.1.1 Сфера застосування.....	30
2.1.2 Функції системи	34
2.1.3 Джерело початкових даних	34
2.1.4 Результати, вихідні дані.....	34
2.2 Вимоги до обчислювального середовища	35
2.3 Якість системи	35
2.3.1 Виконання стандартів та узгодження	38
2.3.2 Можливість перенесення із точки зору апаратного та обчислюваного середовища	38
2.3.3 Надійність функціонування.....	38
2.4 Вимоги до оформлення вихідних кодів	38
2.5 Календарний план виконання робіт	44
3 СТРУКТУРА ТА РОЗРОБКА	46
3.1 Розробка серверної частини.....	46
3.1.1 Загальний опис алгоритму.....	46
3.1.2 Розробка структури бази даних.....	50
ВИСНОВКИ	56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	57
ДОДАТКИ	59

					ЧДТУ.191547.012 ПЗ				
Зм	Лист	№ докум.	Підп.	Дата	Система тестування рівня кваліфікації працівника: Back-end WEB-додатку. (Пояснювальна записка)	Лит	Лист	Листів	
Розроб.	Линник В.Ю.								
Перевірив	Плакасова Ж.М.						3	59	
						ФІТІС, кафедра ПЗАС, ПЗ-154			
Н. контр.	Півень О.Б								
Затверд.	Первунінський С.М.								

СПИСОК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

JS	–	Javascript
API	–	Application Programming Interface
IDE	–	Integrated Development Enviroment
HTML	–	Hyper Text Markup Language
CSS	–	Cascading Style Sheets
IT	–	Інформаційні технології
CRUD	–	Create Read Update Delete
БД	–	База даних
ОС	–	Операційна система
ПЗ	–	Програмне забезпечення
СУБД	–	Система управління базами даних
CSV	–	Comma-Separated Values
JSON	–	JavaScript Object Notation
HTTP	–	Hyper Text Transfer Protocol
SMTP	–	Simple Mail Transfer Protocol

ВСТУП

Що таке інформаційні технології? Люди, які не пов'язані з ІТ, швидше за все, скажуть, що це щось складне, незрозуміле й розумне. Але чи так це насправді? Для аборигена, що не вміє читати й писати, навіть найпростіше речення буде непосильним, але коли ти володієш мовою – ти легко можеш донести навіть найскладніші думки простими й зрозумілими словами. Так само й з мовами програмування. Адже, не потрібно бути генієм, щоб навчитися читати й писати! Простими словами, інформаційні технології – це все, що пов'язано з обробкою, зберіганням і передачею даних. На сьогоднішній день, технології розвиваються досить стрімко, і все більше впливають на наше життя.

Поки помилки у програмах не приносили відчутних неприємностей їх користувачам, програмісти завдавали шкоди тільки собі, і тільки вони були зацікавлені підвищувати свою кваліфікацію. Ситуація змінилася. Програмне забезпечення почало допомагати в управлінні компаніями з мільйонними оборотами, технологічними процесами, життєво важливими системами.

Отже, зросла зацікавленість суспільства у створенні ПЗ без суттєвих помилок, що вимагає від програмістів підвищеної відповідальності і достатнього професійного рівня.

Актуальність даної теми полягає у тому, що компаніям потрібно перевіряти рівень працівників, які в них працюють, не витрачаючи багато часу на різні опитування. Також, даний web-додаток полегшить та спростить процедуру співбесіди та визначення рівня працівника.

Метою роботи є систематизація здобутих теоретичних знань та використання їх на практиці. Тому, було вирішено розробити web-додаток, яким будуть користуватись компанії. Вони зможуть з легкістю додати в додаток свої бібліотеки питань та відповідей до них, та використовувати їх для опитування різного рівня працівників. Як результат кожен додаток буде мати свою базу даних, та свої унікальні питання, тому що, як ми знаємо, рівень оцінювання працівників у кожного свій.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Характеристика предметної області

У сучасному світі все пов'язано з ІТ технологіями. Щороку кількість працівників ІТ-компаній, а також осіб, що працюють в цій галузі зростає. Також, змінюються і завдання, які повинні вирішувати люди. Більшість проблем які могла вирішити одна людина, уже не актуальні. Робота в команді, навички до самоосвіти та велика кількість досвіду, ось декілька характеристик, які потрібні компаніям.

У зв'язку з вище зазначеними факторами, прийнято рішення про створення web-додатку, який буде оцінювати досвід та рівень людей, які хочуть працювати в ІТ сфері.

1.2 Аналіз існуючих аналогів

Перший додаток, який було розглянуто є JSEHELPER. JSEHELPER – це блог, на якому користувачі можуть розглянути лист популярних питань, для співбесіди на різні позиції роботи, та відповіді до них.

Також є можливість зв'язку з розробниками, для того щоб запропонувати свою категорію, або сповістити про проблему, як що таку було знайдено.

При переході на головну сторінку, можемо бачити зручне меню, та групування питань по категоріям.

При переході на певну категорію, відкривається сторінка цієї категорії, меню по ній та список всіх категорій на які ми можемо перейти.

Недоліки додатку:

- відсутність авторизації;
- немає можливості пройти тест;
- відсутність сортування;
- відсутність статистики питань;
- немає можливості створення питань;
- відсутність редагування;
- відсутність будь-якої статистики;
- відсутність зміни мови.

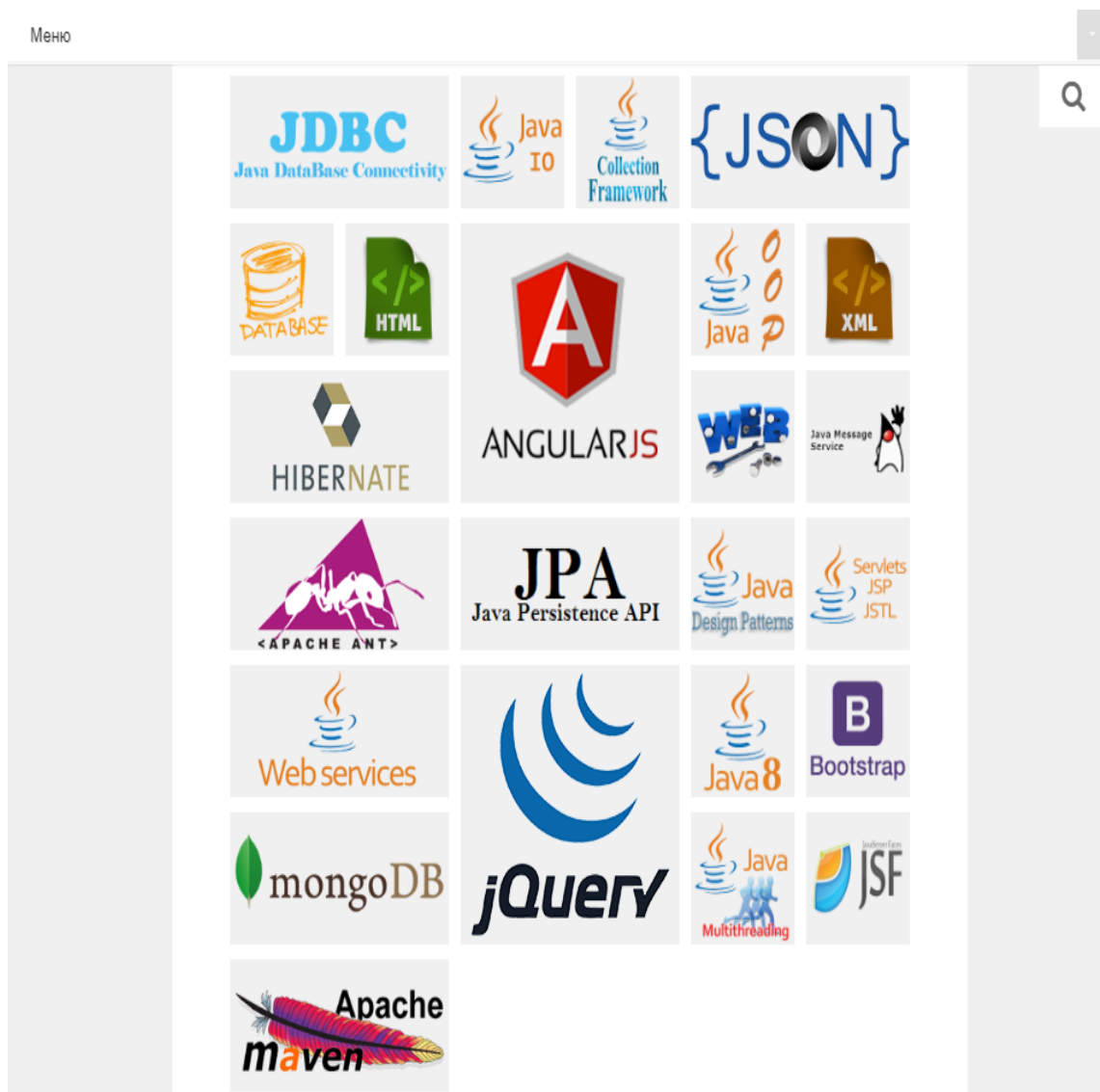


Рисунок 1.1 – Головна сторінка JSEHELPER

ANGULARJS FRONT-END DEVELOPER OTHER INTERVIEW

Ответы на вопросы на собеседование Основы AngularJS.

👤 VASYL1889 ⌚ 15:24:00 💬 2 КОММЕНТАРИЕВ

Что такое AngularJS? AngularJS – структурированный JavaScript-фреймворк с открытым исходным кодом для динамических web-приложений. Предназначен для разработки одностраничных приложений. Позволяет использовать HTML в качестве языка шаблонов, а так же расширять HTML-синтаксис, чтобы код

ДАЛЬШЕ »

🐦 8+ in vk f @

Рисунок 1.2 – Категорія AngularJS в JSEHELPER

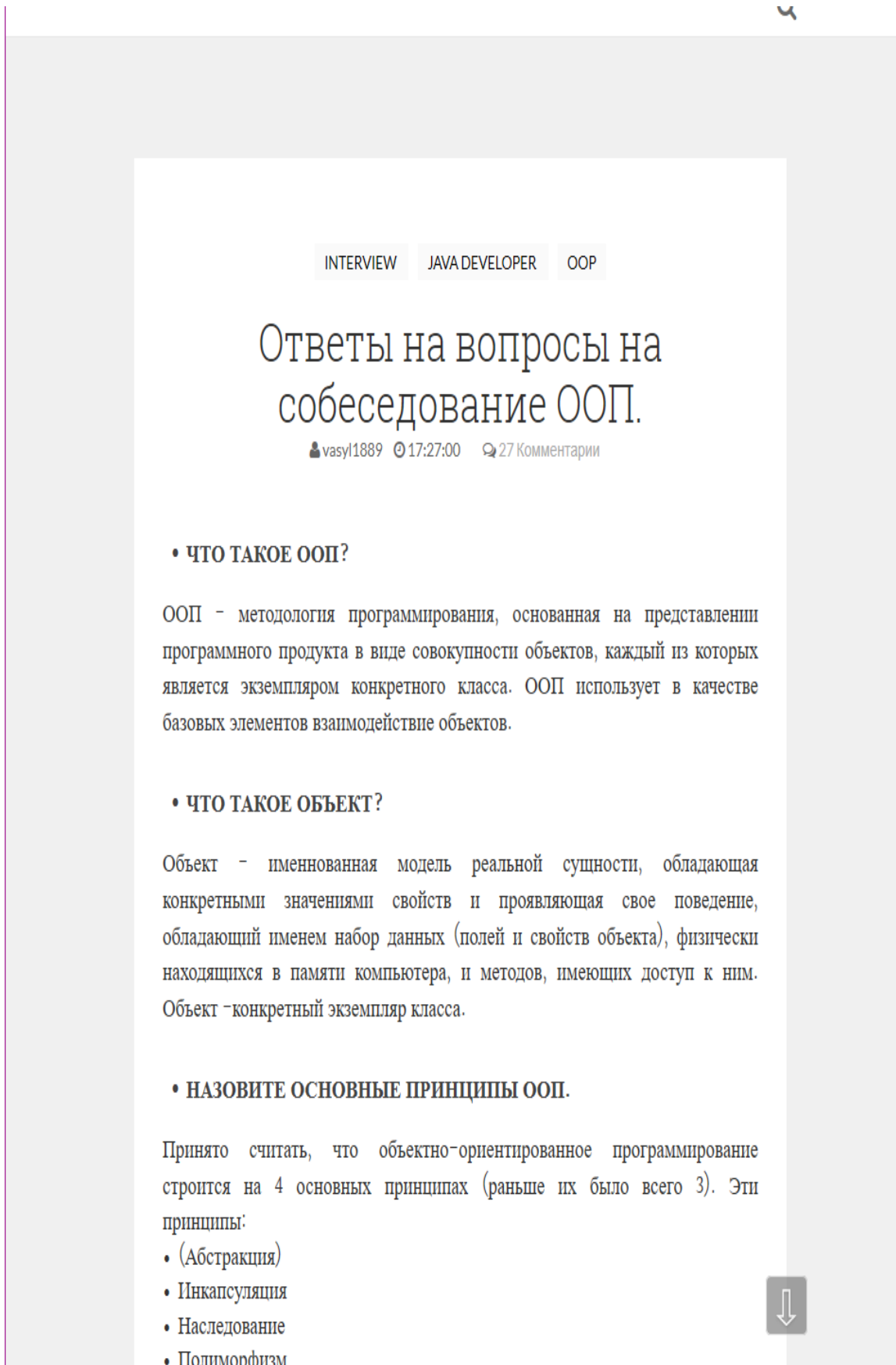


Рисунок 1.3 – Категория OOP в JSEHELPER

Наступним рішенням, що було розглянуто є Quizful. Quizful – безкоштовний сервіс онлайн тестування IT спеціалістів.

Реєстрація на ресурсі Quizful дозволяє проходити тести різних рівнів складності, дивитися правильні відповіді з поясненнями після їх проходження, обговорювати з іншими користувачами статті, питання тестів і питання співбесід.

Ви можете обговорювати питання з іншими користувачами, та проходити тестування разом з друзями, змагаючись за кращий результат.

На данному ресурсі, ви також можете отримати сертифікат, який засвідчує пройдений тест.

Недоліки додатку:

- відсутність сортування;
- відсутність створення питань;
- велика кількість реклами.

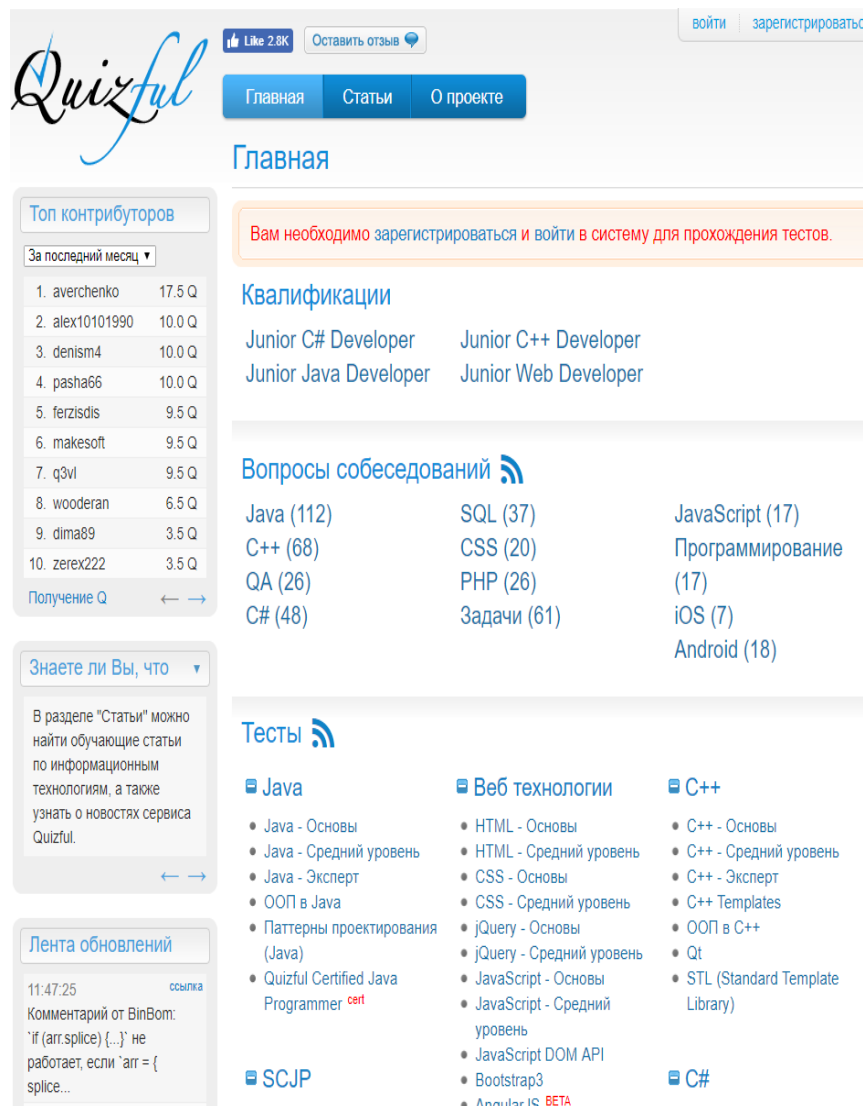


Рисунок 1.4 – Головна сторінка Quizful

Даний додаток набагато функціональніший, ніж попередній. З його допомогою, ви можете пройти тести які вам потрібно, і до багатьох спеціальностей.

Також, в додатку присутні статистика пройдених тестів, та можливість коментарів або обговорення.

Quizful Like 2.8K Оставить отзыв

Главная Статьи О проекте

JavaScript / В чём опасность использования for .. in для объекта или массива?

Автор: JokerNN

Вопрос

Возьмём следующий код

```
var arr = [3, 4, 5];
for (var i in arr){
  console.log(i);
}
```

Что выведется в консоль? Какие могут с ним возникнуть проблемы?

Ответ показать

Если Вам понравился вопрос, проголосуйте за него

Голосов: 67 Голосовать

Комментариев: 7 ↑ обновить

morteryler 27.03.2018 | 08:46:10 #
Лучше использовать for .. of тогда не нужно будет проверку делать
ответить

qwertyasd456 09.08.2017 | 12:18:50 #
спасибо за информацию
ответить

khryshyn 03.08.2016 | 23:14:48 #

Топ контрибуторов

За последний месяц ▼

1. averchenko	17.5 Q
2. alex10101990	10.0 Q
3. denism4	10.0 Q
4. pasha66	10.0 Q
5. ferzisdiz	9.5 Q
6. makesoft	9.5 Q
7. q3vl	9.5 Q
8. wooderan	6.5 Q
9. valeriamarg	3.5 Q
10. zerex222	3.5 Q

Получение Q ← →

Знаете ли Вы, что ▼

Если у вас есть уникальная статья и вы хотите, чтобы она стала достоянием общественности, вы можете разместить ее на Quizful.

← →

Рисунок 1.5 – Питання до javascript в Quizful

Незважаючи на велику кількість матеріалу та функціонал, суттєвим недоліком є те, що ви не маєте можливості створити тест. Також, під час перегляду даного додатку, було виявлено проблеми з дизайном та помилки в тестах.

Третім аналогом є додаток, що має назву Proghub. ProgHub - це молодий проект, де ви можете перевірити свій рівень знань за допомогою тестування з різних мов програмування.

Головна відмінність від інших аналогів це те, що проект підтримує велика кількість людей.

Під час переходу на головну сторінку, ми можемо бачити категорії тестів, та статистику по кожному з них (рисунок 1.5). Як що ми перейдемо на сторінку тестів (рисунок 1.6), ми маємо можливість відсортувати по певній категорії або рівню, всі наявні тести.

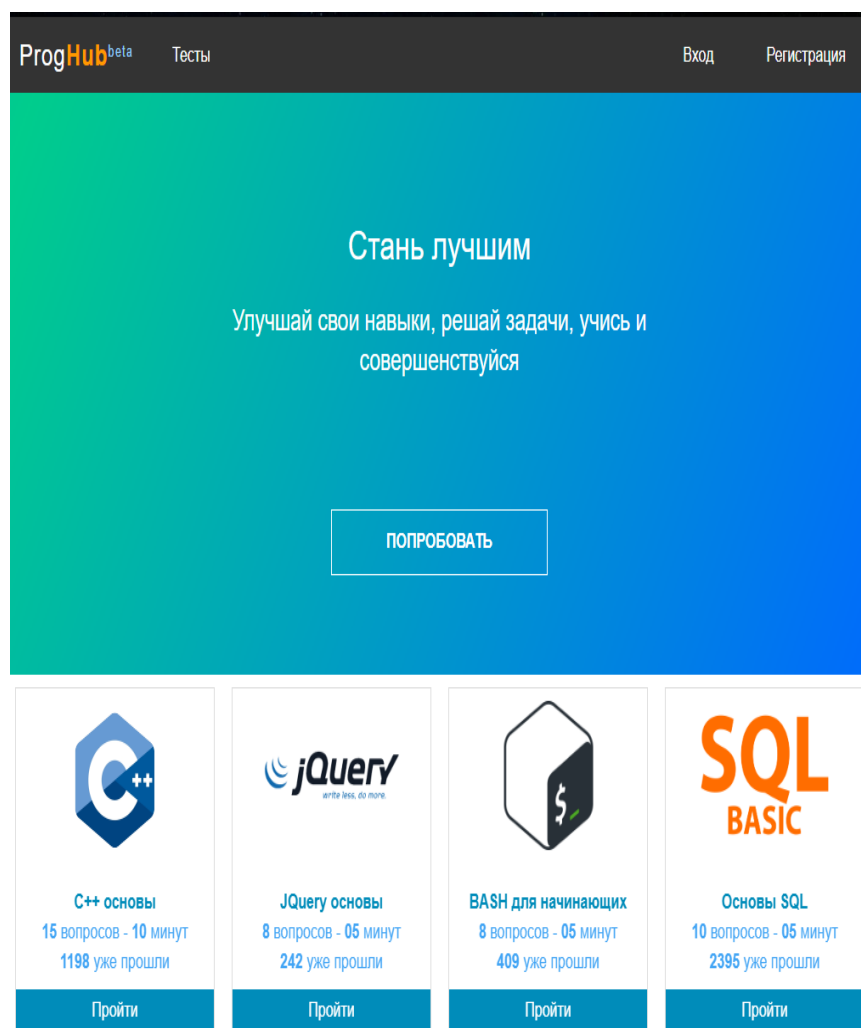


Рисунок 1.6 – Головна сторінка Proghub

ProgHub^{beta}

Тесты

Вход

Регистр

Категория

C++

Qt

PHP

Laravel

JS

Jquery

C#

CSS

C

HTML

Java

Python

OOP

Database

SQL


Go

Уровень

junior


middle

senior




Go основы
10 вопросов - 10 минут
242 уже прошли

Пройти




BASH для начинающих
8 вопросов - 05 минут
409 уже прошли

Пройти




Основы SQL
10 вопросов - 05 минут
2395 уже прошли

Пройти




ООП в Java
10 вопросов - 10 минут
2003 уже прошли

Пройти




ООП в PHP
8 вопросов - 05 минут
292 уже прошли

Пройти




Основы Spring Framework
8 вопросов - 05 минут
233 уже прошли

Пройти




Django основы
8 вопросов - 05 минут
347 уже прошли


Пройти




C++ для начинающих
8 вопросов - 05 минут
1565 уже прошли

Пройти








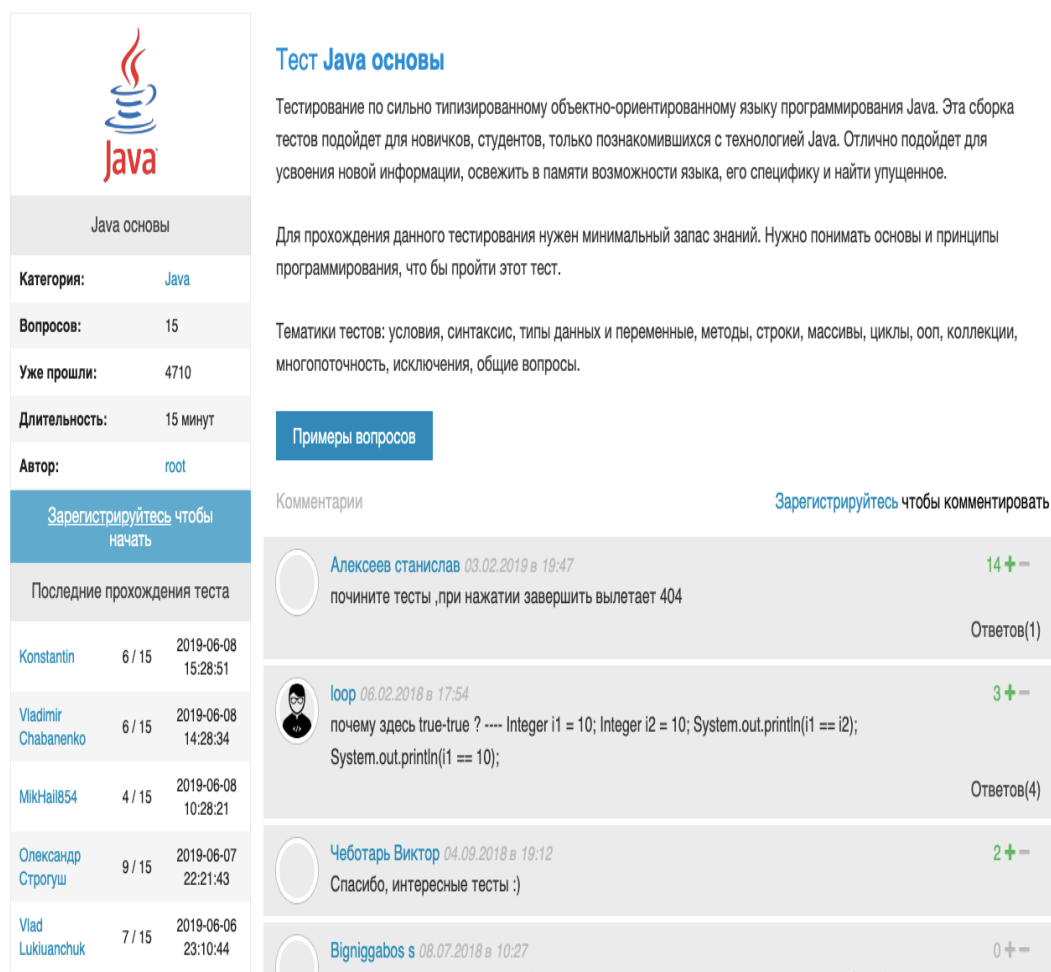


Рисунок 1.7 – Сторінка тестів Proghub

Основним недоліком цієї системи є відсутність можливості створення тесту. Окрім цього слід зазначити, що даний додаток має тести, тільки для ознайомлення з спеціальністю, а не підготовкою до співбесіди.



Тест Java основы

Тестирование по сильно типизированному объектно-ориентированному языку программирования Java. Эта сборка тестов подойдет для новичков, студентов, только познававшихся с технологией Java. Отлично подойдет для усвоения новой информации, освежить в памяти возможности языка, его специфику и найти упущенное.

Для прохождения данного тестирования нужен минимальный запас знаний. Нужно понимать основы и принципы программирования, что бы пройти этот тест.

Тематики тестов: условия, синтаксис, типы данных и переменные, методы, строки, массивы, циклы, ооп, коллекции, многопоточность, исключения, общие вопросы.

[Примеры вопросов](#)

Комментарии

[Зарегистрируйтесь чтобы комментировать](#)

Алексеев станислав 03.02.2019 в 19:47 14 +
 почините тесты ,при нажатии завершить вылетает 404
 Ответов(1)

loop 06.02.2018 в 17:54 3 +
 почему здесь true-true ? ---- Integer i1 = 10; Integer i2 = 10; System.out.println(i1 == i2);
 System.out.println(i1 == 10);
 Ответов(4)

Чеботарь Виктор 04.09.2018 в 19:12 2 +
 Спасибо, интересные тесты :)

Bigniggabos s 08.07.2018 в 10:27 0 +

Рисунок 1.8 – Сторінка тесту java в Proghub

Жодне з перерахованих вище рішень повною не виконує повною мірою поставлену задачу, і тому є доцільною розробка власного програмного продукту.

1.3 Обґрунтування обраних предметних рішень

Під час реалізації програми дипломної роботи було обрана низка актуальних для предметної області програмних рішень. А саме:

1 Проект є інтелектуальною системою, яка складається з 2 частин — інтерфейс користувача написаний на мові програмування Angular, та сервер для обробки даних який написаний на Node.

NodeJs — це платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript.

Раніше ви могли запустити JavaScript тільки в браузері, але одного разу розробники розширили його, і тепер ви можете запускати JS на своєму комп'ютері в якості окремого додатка. Так з'явився Node.js.

Тепер ви можете зробити набагато більше з JavaScript, ніж просто інтерактивні веб-сайти.

Тепер у JavaScript є можливість робити те, що можуть робити інші скриптові мови програмування, такі як Python.

Обидва - браузерні JavaScript і Node.js запускаються в середовищі виконання V8. Цей движок використовує ваш JS код, і перетворює його в більш швидкий машинний код. Машинний - низькорівневий код, який комп'ютер може запускати без необхідності спочатку його інтерпретувати.

Переваги платформи Node:

- здатність тримати багато «наскрізних» реквестів. У найпоширенішому випадку, сервер не чекає, коли БД відповість, а обробляє ще інші реквести, те ж саме з I / O, повідомленнями / реквестами, заливанням файлів на S3 і т.і;
- дозволяє розбивати програми на модулі, що сприяє повторному використанню коду в інших програмах;
- потоки володіють великою силою: в вашому розпорядженні асинхронність в роботі з введенням і виведенням і ви можете перетворювати дані в незалежних етапах;
- масштабованість закладена в дизайні. Ви ніколи не будете запускати 1 процес Node.js;
- NPM – великий пакетний менеджер;
- можливість писати ізоморфні універсальні додатки, що працюють на клієнті і на сервері, що абстрагує front-end і back-end;
- займає мало пам'яті. Як згадано вище, дуже допомагає в розгортанні мікросервісів;

- велика кількість готових бібліотек які можна використовувати в своєму додатку;
- як що писати код асинхронним методом, в декілька раз збільшиться продуктивність роботи додатку, тому що програма не буде чекати виконання інших функцій.

Як працює Event Loop в JavaScript.

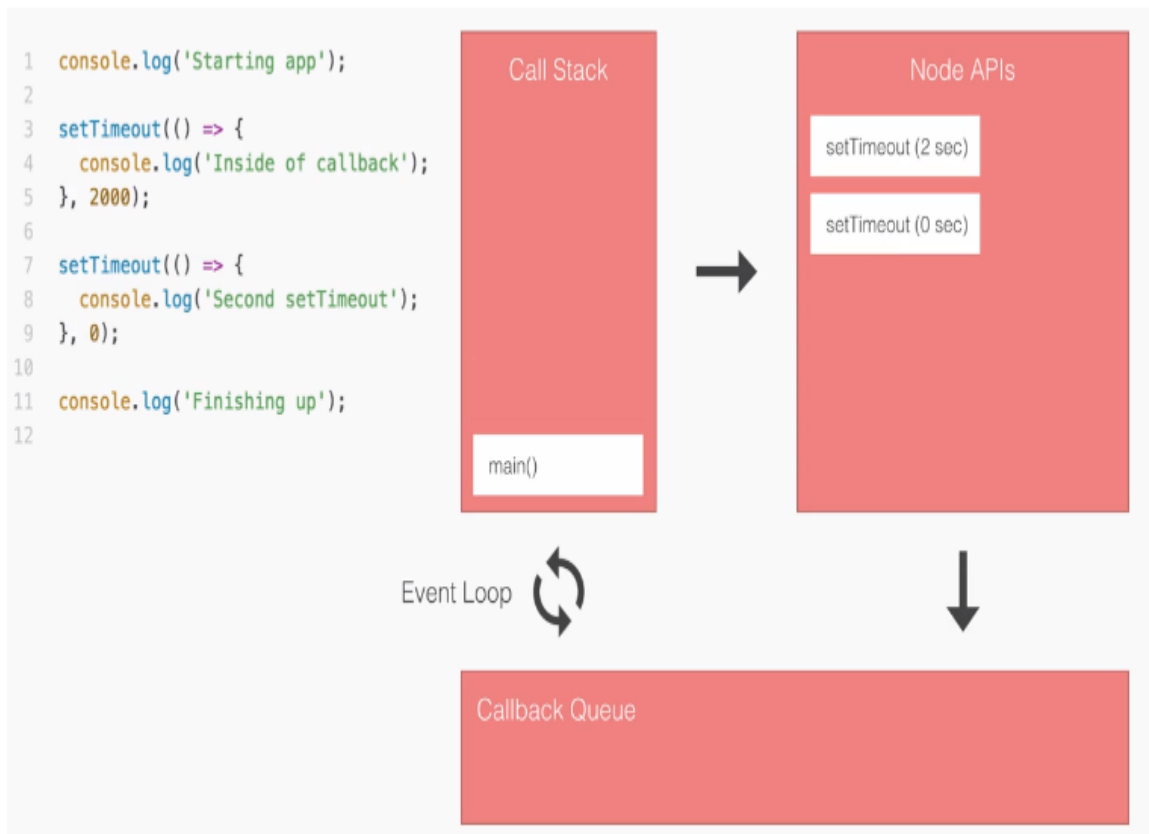


Рисунок 1.9 - Event Loop JavaScript

- посилаєте main() в стек викликів;
- посилаєте console.log() в стек викликів. Він запускається, і відразу з'являється;
- посилаєте setTimeout (2000) в стек. setTimeout (2000) це - Node API. Коли ми його викликаємо, ми реєструємо пару подія-коллбек. Подія буде чекати 2000 мілісекунд, а потім викличе коллбек;
- після реєстрації, setTimeout (2000) з'являється в стеці викликів;
- другий setTimeout (0) реєструється таким же чином. Тепер у нас є два API-інтерфейсу Node, які очікують виконання;

- після очікування 0 секунд `setTimeout (0)` переміщається в чергу виконання коллбеків (`callback queue`), і те ж саме відбувається з `setTimeout (2000)`;
- у черзі виконання коллбеків, функції чекають, коли стек викликів буде порожнім, тільки одна функція може виконуватися одночасно. Це забезпечує `event loop`;
- викликається остання `console.log()`, а `main()` викликається з стека викликів;
- цикл подій бачить, що стек викликів порожній, а черги зворотного виклику - немає. Таким чином, він переміщує зворотні виклики в стек викликів для виконання.

Тобто, JavaScript – унікальна мова програмування. Ми можемо писати додатки на одній мові, використовуючи складні архітектурні рішення з `front-end` і `back-end` частинами.

2 Під час розробки програмного продукту використовувався NPM.

NPM – це пакетний менеджер для Node. Він містить в собі тисячі бібліотек, які спрощують і полегшують процес написання програмного продукту.

node-sass

4.12.0 • Public • Published a month ago

Readme 17 Dependencies 5,957 Dependents 135 Versions

Supported Node.js versions vary by release, please consult the [releases page](#). Below is a quick guide for minimum support:

NodeJS	Minimum node-sass version	Node Module
Node 12	4.12+	72
Node 11	4.10+	67
Node 10	4.9+	64
Node 8	4.5.3+	57

install

```
> npm i node-sass
```

± weekly downloads

3,429,962

version 4.12.0 license MIT

open issues 131 pull requests 33

homepage repository

Рисунок 1.10 – NPM пакет node-sass

Особливості NPM:

- версії і назви залежностей зберігає в файл `package.json`;
- всі бібліотеки зберігаються в `node_modules`;
- швидкість роботи;
- велика кількість розробників;
- один з найбільших пакетних менеджерів, з готовими бібліотеками для використання з мовою javascript;
- гнучкість;
- динамічне поширення своїх розробок;
- зручне склеювання з іншими бібліотеками.

3 Для відправлення повідомлень на пошту, була обрана бібліотека Nodemailer.

Nodemailer - це модуль для додатків Node.js, що дозволяють легко пересилати електронну пошту. Проект розпочався ще в 2010 році, коли відсутній нормальний варіант надсилання повідомлень електронної пошти, сьогодні це рішення, до якого більшість користувачів Node.js переходять за замовчуванням.

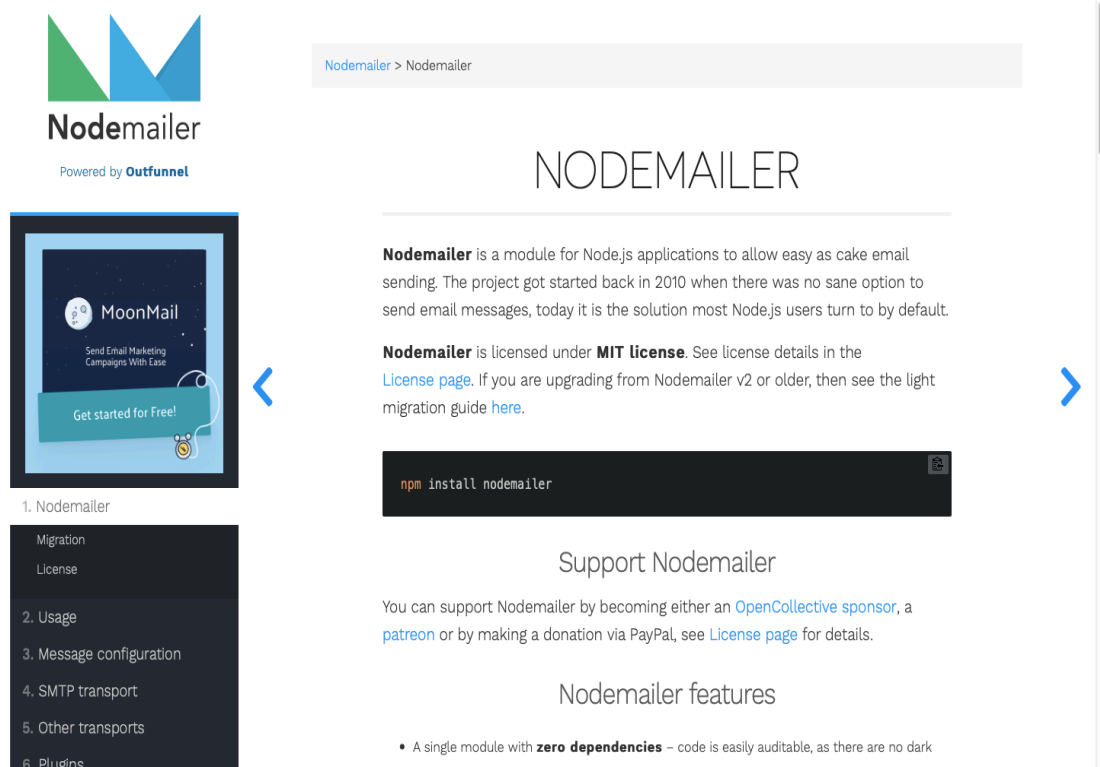


Рисунок 1.11 – офіційна сторінка Nodemailer

Переваги використання Nodemailer:

- один модуль з нульовими залежностями - код легко перевіряється;
- фокус на безпеці;
- підтримка Unicode для використання будь-яких символів, включаючи підтримку emoji Windows;
- ви можете встановити його з npm у Windows, як і будь-який інший модуль, відсутні компільовані залежності. Використовуйте його без проблем з Azure або у вікні Windows;
- використовуйте вміст HTML, а також альтернативний варіант звичайного тексту;
- додати додатки до повідомлень;
- вкладені зображення вбудованого вмісту для HTML та ваш дизайн не блокується;
- підтримка користувацьких плагінів для маніпулювання повідомленнями;
- OAuth2 аутентифікація;
- проксі для підключень SMTP.

4 Для збереження результатів аналізу зображень використовувалась база даних PostgreSQL.

PostgreSQL не просто реляційна, а об'єктно-реляційна СУБД. Це дає йому деякі переваги над іншими SQL базами даних з відкритим вихідним кодом, такими як MySQL, MariaDB і Firebird.

Фундаментальна характеристика об'єктно-реляційної бази даних - це підтримка об'єктів і їх поведінки, включаючи типи даних, функції, операції, домени і індекси. Це робить Постгрес неймовірно гнучким і надійним. Серед іншого, він вміє створювати, зберігати та видавати складні структури даних.

Одним з фундаментальних понять POSTGRES є class. Class є іменована колекція примірників (instances) об'єктів. Кожен екземпляр має колекцію іменованих атрибутів і кожен атрибут має певний тип. Класи можуть бути трьох типів - це основний клас, чиї екземпляри зберігаються в базі даних,

віртуальний (view), чий екземпляр матеріалізується тільки при запиті (вони підтримуються системою керування правилами), і може бути версією іншого (parent) класу.

Для налаштування POSTGRES було використано pgAdmin. PgAdmin - це програмне забезпечення, що забезпечує графічний інтерфейс для роботи з базовими даними та полегшує роботу.



Рисунок 1.12 – Інтерфейс PGADMIN

PostgreSQL підтримується на всіх сучасних Unix системах (34 платформи), включаючи найбільш поширені, такі як Linux, FreeBSD, NetBSD, OpenBSD, SunOS, Solaris, DUX, а також під Mac OS X. Починаючи з версії 8.X PostgreSQL працює в 'native' режимі під MS Windows NT, Win2000, WinXP, Win2003. Відомо, що є успішні спроби працювати з PostgreSQL під Novell Netware 6 і OS2.

Переваги PostgreSQL:

- надійність PostgreSQL є перевіреним і доведеним фактом і забезпечує повну відповідність принципам ACID - атомарність, несуперечливість, ізолюваність, збереження даних;

- багатоверсійність (Multiversion Concurrency Control, MVCC) використовується для підтримки узгодженості даних в конкурентних умовах, в той час як в традиційних базах даних використовуються блокування. MVCC означає, що кожна транзакція бачить копію даних (версію бази даних) на час початку транзакції, незважаючи на те, що стан бази могло вже змінитися. Це захищає транзакцію від неузгоджених змін даних, які могли бути викликані (інший) конкурентної транзакцією, і забезпечує ізоляцію транзакцій. Основний вииграш від використання MVCC в порівнянні з блокуванням полягає в тому, що блокування, яке ставить MVCC для читання не конфліктує з блокуванням на запис, і тому читання ніколи не блокує запис і навпаки. Конкурентні операції записи 'заважають' один одному тільки при роботі з однієї і тієї ж записом;
- наявність Write Ahead Logging (WAL) - загальноприйнятий механізм протоколювання всіх транзакцій, що дозволяє відновити систему після можливих збоїв. Основна ідея WAL полягає в тому, що всі зміни повинні записуватися в файли на диск тільки після того, як ці записи журналу, що описують ці зміни будуть і гарантовано записані на диск. Це дозволяє не скидати сторінки даних на диск після фіксації кожної транзакції, так як ми знаємо і впевнені, що зможемо завжди відновити базу даних використовуючи журнал транзакцій;
- Point in Time Recovery (PITR) - можливість відновлення бази даних (використовуючи WAL) на будь-який момент в минулому, що дозволяє здійснювати безперервне резервне копіювання кластера PostgreSQL;
- реплікація також підвищує надійність PostgreSQL. Існує кілька систем реплікації, наприклад, Slony (тестується версія 1.1), який є вільним і найбільш використовуваним рішенням, підтримує master-slaves реплікацію. Очікується, що Slony-II буде підтримувати multi-master режим;

- цілісність даних є серцем PostgreSQL. Крім MVCC, PostgreSQL підтримує цілісність даних на рівні схеми - це зовнішні ключі (foreign keys), обмеження (constraints).

5 Для тестування веб-запитів до серверу використовувався REST-клієнт Postman. Клієнт Postman - це дуже популярний і простий у використанні компонент HTTP запитів.

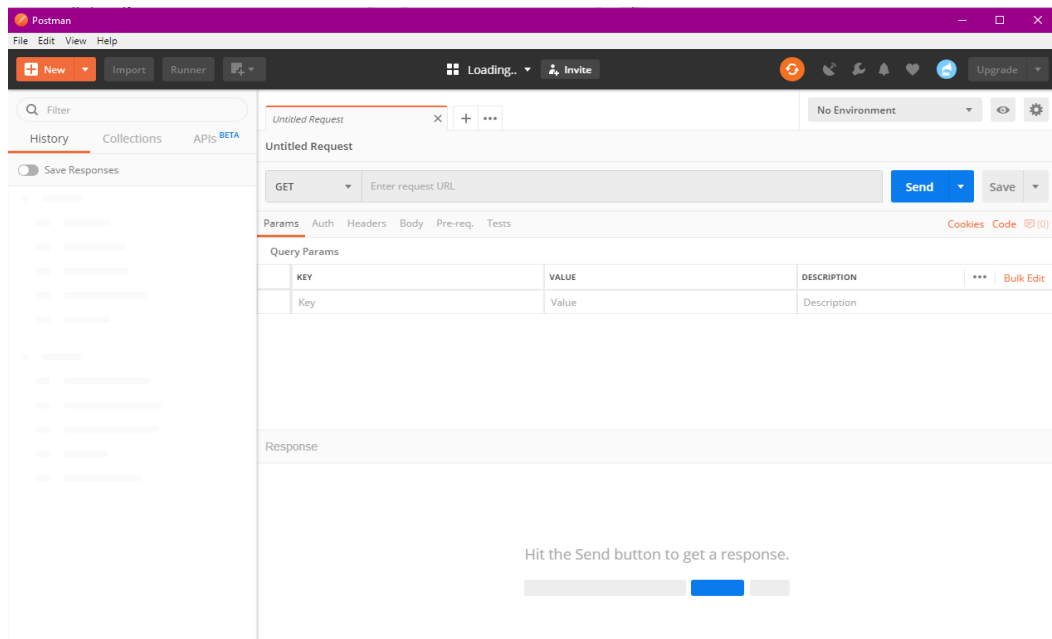


Рисунок 1.13 – Інтерфейс Postman

Postman – дуже зручний в використанні клієнт, для тестування запитів з сервера. Потрібно записати всі параметри, які потрібні для запиту.

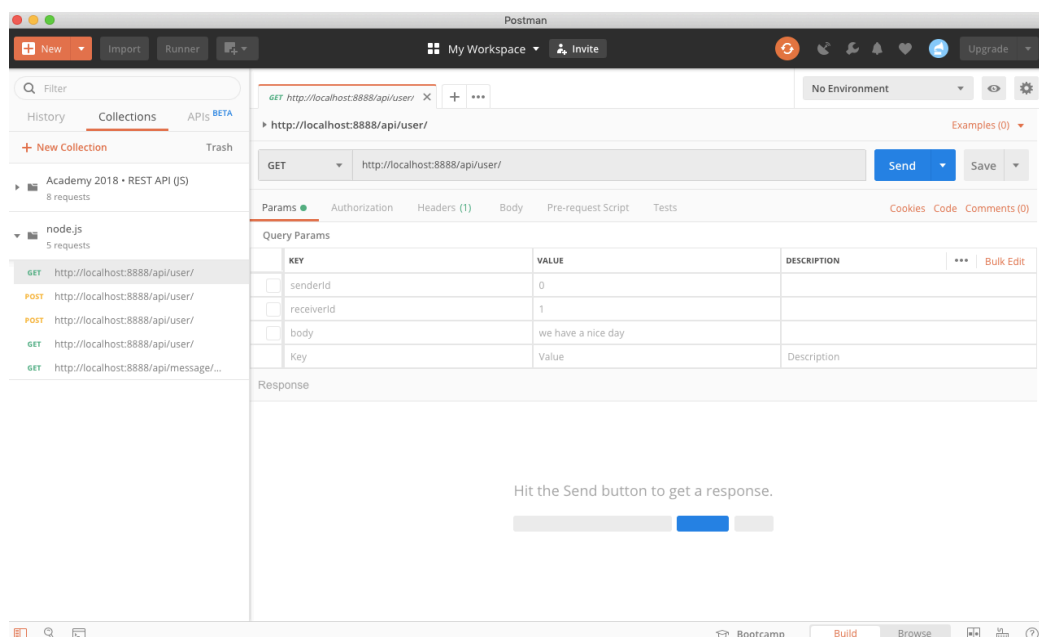


Рисунок 1.14 – Приклад запиту Postman

Ми ставимо URL в змінну оточення і використовуємо його щоб прописувати кінцеву точку. І на вкладці "тести" потрібно вибрати фрагмент, щоб перевірити стан відповіді. Єдине, що може бути проблемою для початківців, це складність в налаштуванні запитів з токеном або логіном. Але це вирішується знову через введення змінної в навколишнє середовище. І код для цього можна буде приймати в фрагменті з розмовною назвою "встановити змінну середовища".

Переваги використання:

- зручна розробка API. Ми можемо без проблем тестувати CRUD запити, та використовувати параметри до них;
- швидке встановлення, або використання в браузері;
- мінімалізм використання командної панелі, що значно полегшує процес;
- форматування коду;
- кілька способів налаштування авторизації та створення або керування файлами cookie;
- можливість автоматизувати процес;
- при необхідності, є наявність створення змінних, та їх використання в запитах;
- динамічні запити. Також, ви можете запустити колекцію запитів, що будуть виконуватись по черзі;
- зручність використання, і простота в використанні полегшує роботу та робить її не складною;
- зберігання запитів та дій з API;
- легкість в створенні документації;
- налаштування теми;
- зручно створювати макети запитів для команд, щоб імітувати кожну кінцеву точку і відповідні відповіді в колекції. Розробники можуть переглядати потенційні відповіді, не розвертаючи повністю систему,

тому члени команди можуть швидко узгодити результати та знайти помилки в проектуванні на ранніх етапах розробки API.

6 У якості IDE було обрано IntelliJ WebStorm.

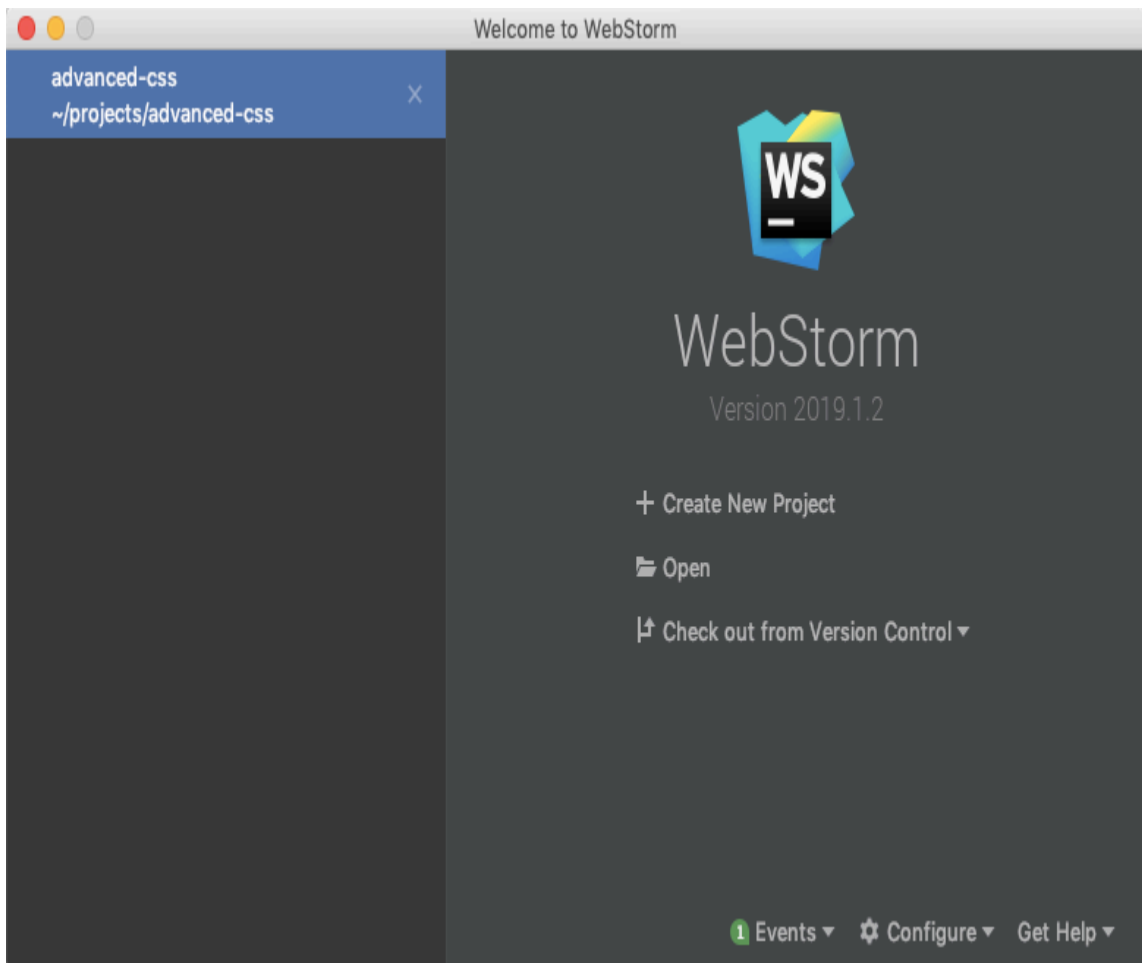


Рисунок 1.15 – Інтерфейс запуску WebStorm

Можливості WebStorm:

- статичний аналіз коду, підсвічування синтаксису і помилок;
- зручний і розумний редактор JavaScript, HTML і CSS, який підтримує також і інші мови, наприклад TypeScript, CoffeeScript, Dart, Less, Sass і Stylus і фреймворки, наприклад, Angular, React і Meteor;
- WebStorm робить розробку проекту простою і зручною, забезпечуючи підсвічування і автодоповнення коду, його аналіз по ходу редагування, швидку навігацію і рефакторинг. Він має потужні інструменти налагодження та інтеграції з системами управління версіями (Git, GitHub, Subversion, Perforce, Mercurial, CVS), розуміє структуру проекту і код, відстежує помилки за допомогою систем

ESLint, JSHint, JSLint, TSLint, Stylelint і пропонує їх рішення. Вбудовані в IDE інструменти для тестування і роботи з проектом допомагають в розробці і роблять її зручніше і продуктивніше;

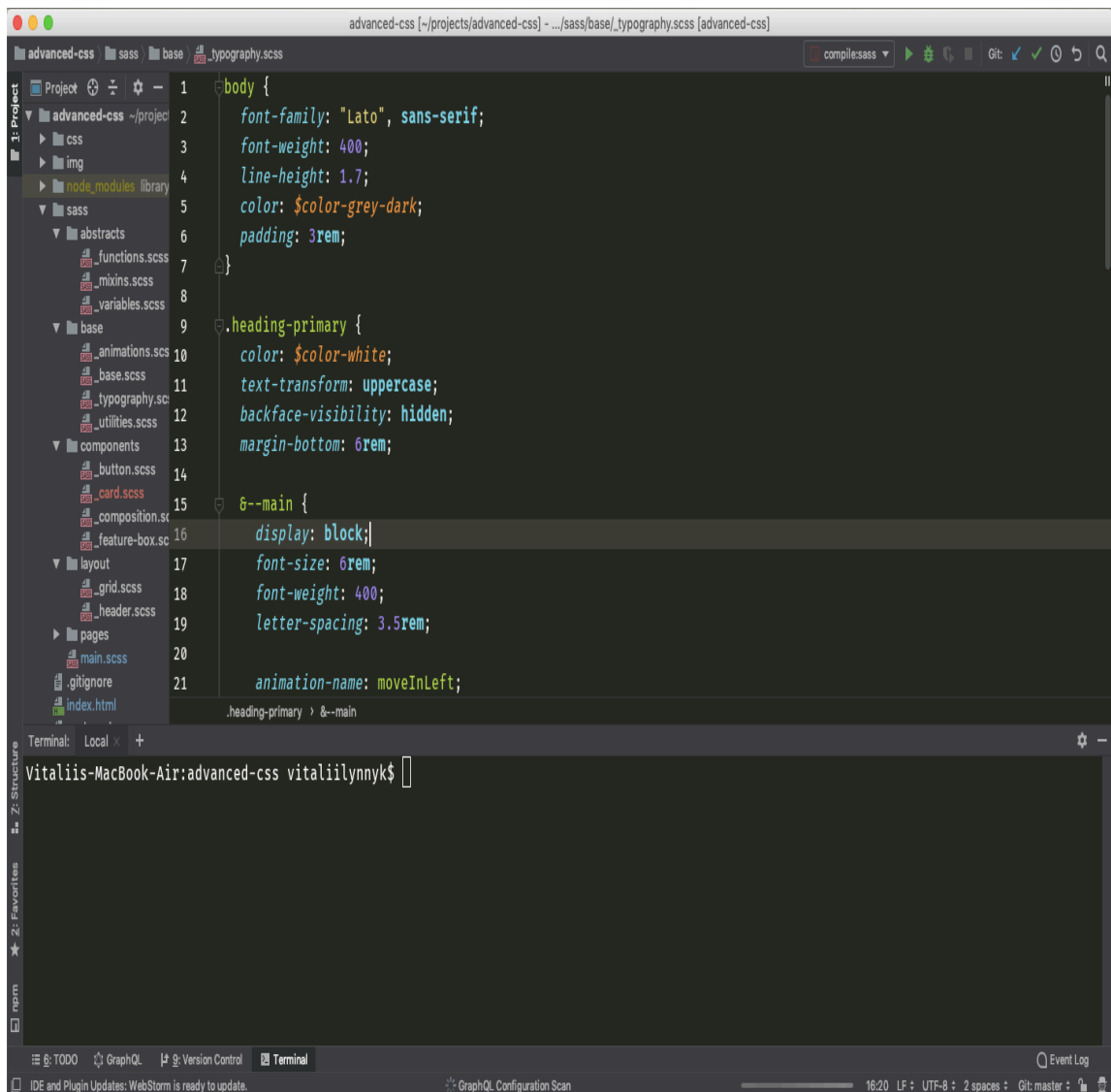


Рисунок 1.16 – Інтерфейс запуску WebStorm

- навігація серед проектів і коду: відображення файлової структури проекту, швидкий перехід між файлами, класами, методами і використаннями методів;
- рефакторинг: перейменування, витяг методу, введення змінної, введення константи, підняття і опускання методу тощо;
- інструменти для веб-розробки з використанням фреймворків;
- вбудовані інструменти для юніт-тестування;

- підтримка систем контролю версій: загальний користувацький інтерфейс для Mercurial, Git, Subversion, Perforce і CVS з підтримкою списків змін та злиття;
- автоматичне доповнення;
- перевірка на помилки коду в run time;
- готові шаблони;
- можливість додавати todo, і налаштовувати їх під себе, та змінювати під час написання коду;
- велика кількість готових плагінів;
- гнучкість;
- підтримка великою кількістю розробників, та активна розробка проекту.

Я хотів, щоб інструмент, був повністю заточений на веб-розробку. Так вийшло, що на додаток до WebStorm є ряд інших IDE, які були відкинуті через різні ситуації, які особисто для мене недолики.

Вибір IDE залежить насамперед від вподобань самого розробника.

7 Також під час розробки проекту була використана система контролю версій Git. Це розподілена система керування версіями файлів та спільної роботи. Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом, також можлива прив'язка коментарів розробників до тегів і комітів.

Віддалений доступ до репозиторіїв Git забезпечується git-daemon (планувальник), SSH або HTTP сервером. TCP-сервіс git-daemon входить у дистрибутив Git і є разом з SSH найпоширенішим і надійним методом доступу. Метод доступу HTTP, незважаючи на ряд обмежень, дуже популярний в контрольованих мережах, тому що дозволяє використання існуючих конфігурацій мережеских фільтрів.

Робота з git дуже відрізняється від роботи з іншими системами контролю джерела.

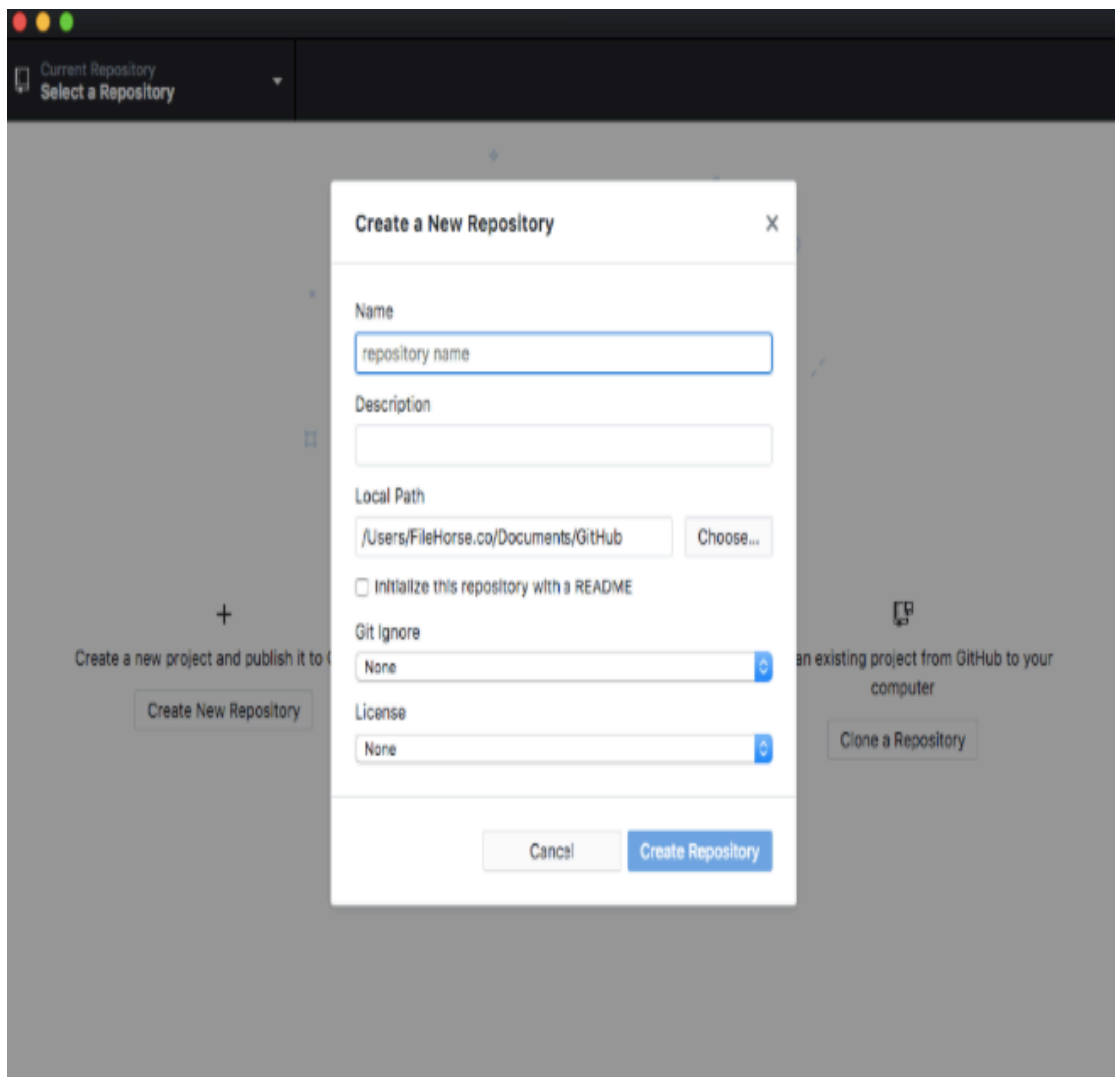


Рисунок 1.17 – Інтерфейс GithubApp

Наявність локального сховища є дуже важливим. Це означає, що ви можете використовувати силу джерела системи контролю (і це багато з git), не втручаючись ні до кого іншого в роботу, та не змінюючи його код. Якщо ваш проект має суперечливу проблему, ви можете просто працювати на ньому, в приватній гілці, використовуючи патчі і поліруючи їх. Таким чином, ви можете повернутися з згладженим набором виправлень. Але навіть в "нормальній роботі" набагато краще, якщо ви можете очистити ваші патчі, перш ніж показувати їх на публіці, і насправді налагодження набагато простіше, якщо у вас є нормальні виправлення, а не тільки "End" знімки.

Це дає вам дуже різні уявлення про вашу роботу. Це дозволяє бачити поточний стан як доповнення до окремих патчів, які створюють історію, яка не просто журнал, але те, що ви поклали там конкретно. Патч комплекти цегли, з яких ви створюєте вашу програму і, при необхідності, рухатися в потрібному напрямку.

Я б ніколи добровільно не повернувся до будь-якої іншої системи контролю версій, я використовував до Git інші системи, але в будь-якій системі контролю версій, яка не підтримує перебазування і місцеві гілки, досить не зручно працювати.

Для операцій з Git-репозиторієм був використаний Git-клієнт SourceTree та термінал windows.

8 Для розгортки веб-серверу був використаний сервіс Heroku[4]. Heroku - хмарна платформа як служба (PaaS), яка підтримує декілька мов програмування та використовується як модель розгортання веб-додатків.

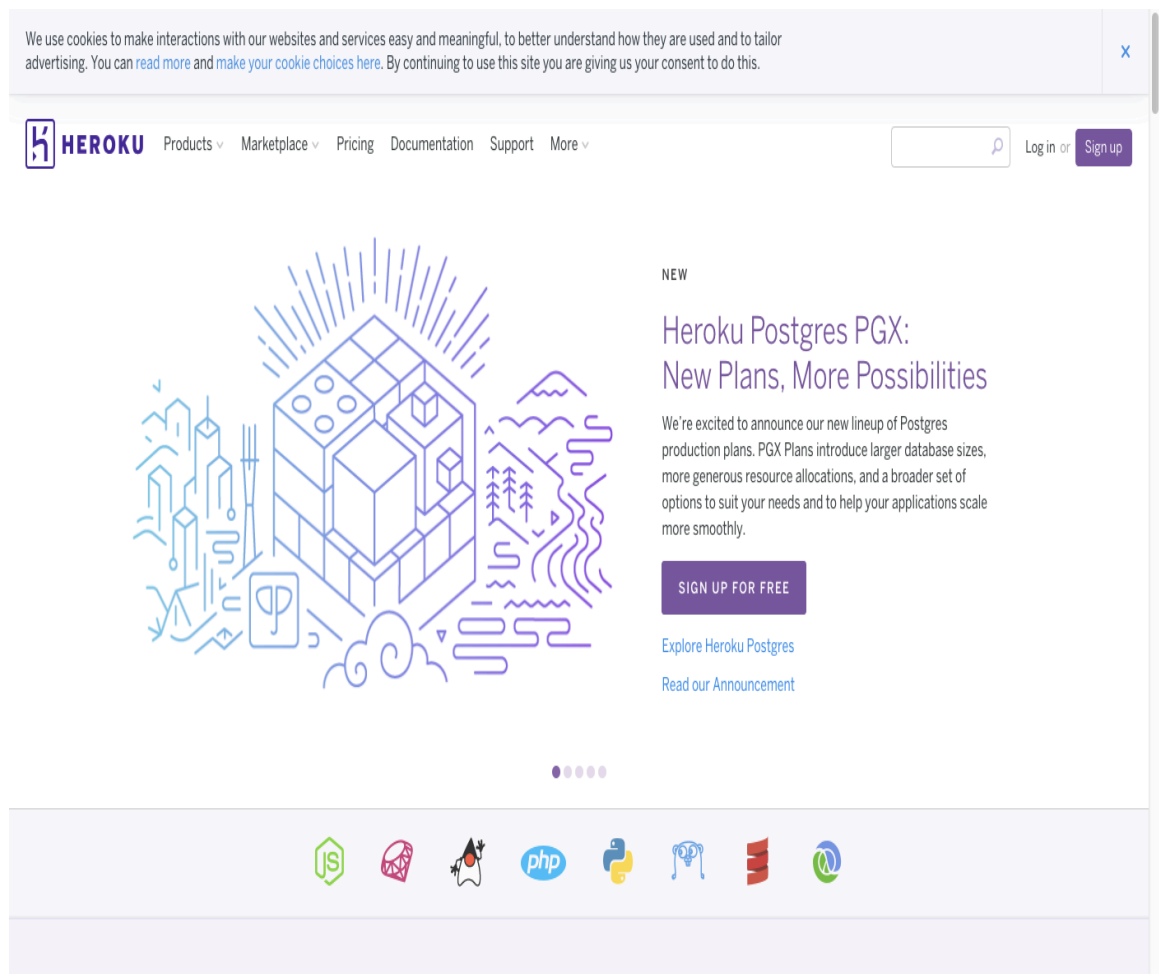


Рисунок 1.18 – Офіційна сторінка Heroku

Heroku підтримує велику кількість мов програмування. Під час розгортки проекту, Heroku створює окремий Git репозиторій. Особливістю Heroku є те, що він автоматично завантажує всі потрібні для проекту бібліотеки, потрібно лише вказати їх у спеціальному файлі. Після завершення розгортки проекту, користувачеві надається окремий домен для взаємодії з розгорнутим програмним засобом. Також Heroku має досить зручну панель керування. Базове безкоштовне рішення, що надається Heroku є достатнім для розгортки простого веб-додатку або API.

Найближчим аналогом до Heroku є один з безлічі сервісів Amazon, а саме EC2. Він надає схожі до Heroku можливості, проте має низку недоліків. По-перше EC2 не має автоматизованої системи розгортки проектів, тому доводиться виконувати це вручну. По-друге віртуальні машини, що надаються EC2 не оновлюються самостійно, тому усі маніпуляції пов'язані з безпекою та актуальністю встановленого образу віртуальної машини лежать на плечах користувача. В кінці кінців базове рішення, що надається EC2 гірше з точки зору характеристик апаратної частини й до того ж платне після першого року використання, на відміну від Heroku.

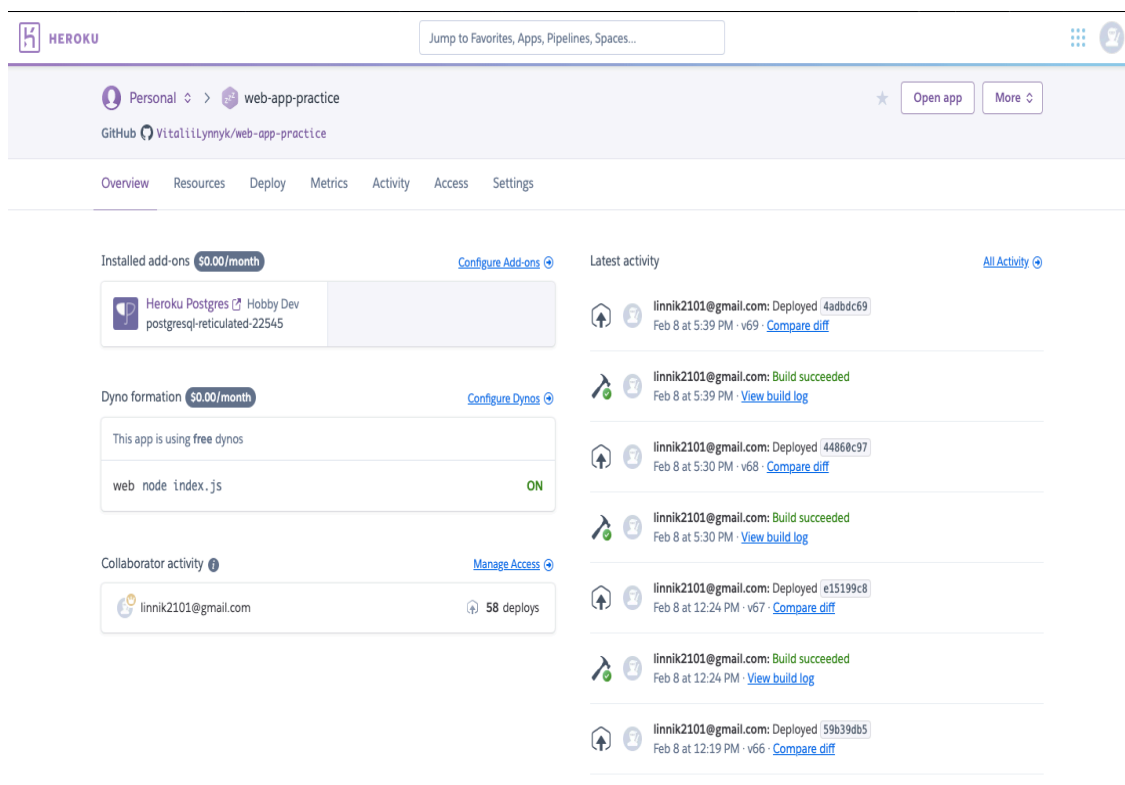


Рисунок 1.19 – Інтерфейс Heroku

9 Для доступу до панелі адміністратора використовувався Google Chrome. Google Chrome — це безкоштовний веб-переглядач, розроблений компанією Google на основі веб-переглядача з відкритим кодом Chromium та іншого відкритого програмного забезпечення.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Загальна характеристика системи

2.1.1 Сфера застосування

Розроблювану систему планується застосовувати фірмах, які займаються розробкою програмного забезпечення, для полегшення набору кваліфікаційних кадрів та спрощення процедури відбору кандидатів на відповідні вакансії.

Той факт, що ви хороший програміст грає напрочуд невелику роль в інтерв'ю. Для того, щоб ваша робота буде продуктивним, ви повинні бути в змозі вирішити величезні, схильні рости проблеми протягом тижнів і місяців. Кожне питання на співбесіді, з іншого боку, триває менше однієї години. Для того, щоб показати себе добре в інтерв'ю, ви повинні знати, як швидко вирішити дрібні проблеми під тиском, при цьому чітко заявивши, ваші думки. Це зовсім інша майстерність (в той же час я не кажу, що здатність легко вирішувати завдання на співбесіді не має ніякого зв'язку з програмними навичками. саме це з'єднання набагато слабкіше, ніж багато компаній вірять, і це інші чинники. Відрізняється від спеціалізованих навичок, у зв'язку з таким вражаючим розбіжність).

Ентузіазм має величезний вплив на результати співбесіди. Близько 50% наших кандидатів, які не проходять подальших співбесід у компаніях, не змогли їх з технічних причин. Компанії зазвичай пояснюють це тим, що людина "не вписується в культуру компанії". Однак, у дев'яти з десяти випадків "вписуватися в культуру", щоб продемонструвати ентузіазм за те, що компанія робить. Компанії наймати людей, які мають жвавий інтерес до своєї місії. Для багатьох, це так само важливо, як технічні знання. І це має сенс. Мотивовані співробітники будуть почувати себе щасливішими і працювати важче.

Інтерв'юерів допомогти кандидатам. Вони дають підказки, реагують на ідеї і, як правило, керівництво інтерв'ю. Але вони не допомагають всім кандидатам таким же чином. Деякі програмісти можуть отримати значну

допомогу таким чином, що людина, яка веде інтерв'ю не буде покласти його в докір. Інші засуджені за будь-які натяки вони отримують. Повірте, ви хочете, щоб допомогти.

Це залежить від того, як розвивається комунікація. Якщо людині подобається ваш підхід, і ви знайдете спільну мову з ним, він буде радий допомогти вам. Ви можете добитися цього, працюючи з конкретним алгоритмом інтерв'ю.

Інтерв'ю на посаду програміста зазвичай складається з питань, пов'язаних безпосередньо з програмуванням, і мова йде про них, що я розповів раніше в цій статті. У той же час, ви можете відповідати на питання про створення конкретної системи. Компанії, схоже, схильні до "мук" таким чином, більш досвідчені кандидати. У цьому випадку потенційний працівник запитує, як вони б розробили складну систему для реальних даних. Прикладами є розробка карт Google, соціальних мереж або, наприклад, створення API для банку.

Перше спостереження: відповідь на питання, пов'язані з розвитком такої системи, вимагає специфічних знань. Ніхто не зажадає від вас створення карт Google (це зайняло багато часу з великою кількістю людей). У той же час, інтерв'юер очікує кандидата на розуміння основних принципів розвитку такої системи. Не хвилюйтеся, як правило, всі такі питання відносяться до Бекенду, так що ви можете зробити значний прогрес, якщо ви розумієте відповідну інформацію добре.

Ви зможете краще виглядати на співбесіді, якщо ви заздалегідь репетирувати відповіді. Це тому, що будь-яка робота інтерв'ю стресових, і стрес негативно впливає на те, як ви навчити себе. Одним із рішень є практика. Інтерв'ю стають менш складними, коли ви придбаєте досвід роботи в бізнесі і в кінцевому підсумку звикнути до них. Ми часто помічаємо, що перші інтерв'ю кандидатів (навіть в тому ж пошуку роботи) є невдалими, але потім вони набувають впевненості і краще виглядати на співбесіді. Немає простого часу для подолання стресу, я рекомендую починати з початку співбесіди. Знайти

список запитань для інтерв'ю (ми рекомендуємо книгу розтріскування кодування інтерв'ю, на російському ринку відомий як "кар'єра програміста. Як отримати роботу в Google, Microsoft або іншої провідної ІТ-компанії») і працювати їх. Встановити таймер, даючи кожне питання 20 хвилин і спробувати відповісти на нього як можна швидше. Практика написання відповідей на дошці (не всі компанії вимагають його, але це найгірший варіант, який варто працювати). Аркуш паперу і перо можуть служити вам імітувати дошку. Якщо у вас є друзі, які можуть допомогти вам практикувати ваше інтерв'ю, то це здорово. Ви можете взяти інтерв'ю один з одним у свою чергу. Читаючи питання, задані в інтерв'ю також дасть вам гарні ідеї, які ви можете скористатися під час реального інтерв'ю. Дивно, багато питань є повторно (в цілому або в частині).

Навіть досвідчені (і стресо-схильні) кандидати виграють від нього. Проходячи інтерв'ю є навик, який значно відрізняється від роботи програміста, і цей навик може атрофія. Але досвідчені програмісти часто (і розумно) відчують, що вони не повинні готуватися до співбесіди. Вони вчаться менше. Тому більш молоді кандидати часто насправді краще відповідати на питання, ніж досвідчені кандидати. Компанії знають про це і, як не парадоксально, деякі говорять нам, що для досвідчених кандидатів, вони встановлення більш низького порогу для відповіді на питання програмування.

Пройшовши інтерв'ю це навичка. Якщо ви великий програміст, штраф, але цього не достатньо. Кожен ніколи не буде інтерв'ю, і належне навчання може допомогти кожному підвищити ймовірність успіху. Ентузіазм є надзвичайно важливим, і навчитися правильно його показувати, ви повинні практикувати. Через відсутність ентузіазму в інтерв'ю, тому що багато програмістів невдачу, оскільки вони не з технічних причин. Інтерв'юери допомагають кандидатам під час співбесіди, і якщо ви правильно вести себе і правильно спілкуватися, то вони вам допоможуть. Практика завжди допомагає. Редінг велику кількість запитань, запитували на інтерв'ю і

самостійного навчання стрес досвідчених в інтерв'ю забезпечить вам успіх і отримати більше пропозицій роботи.

Це не ідеальна ситуація. Підготовка до інтерв'ю дає свої плоди, але навчання програмістів навички, які не відносяться до створення якісного програмного забезпечення, проводить час кожного учасника, що беруть участь у процесі найму. Компанії повинні поліпшити процес інтерв'ювання, зменшуючи вплив академічних знань в галузі інформатики, запам'ятали факти і репетирували дії. Але реальний стан речей тепер відрізняється від того, як ми прагнемо, тому програмісти повинні бути в змозі належним чином підготуватися до інтерв'ю.

Крім того, люди, які інтерв'ю часто роблять це в непрофесійно і кілька відсутніх чином налаштованих (вони воліють програмування замість співбесіди) і, отже, задавати питання, пов'язані з фактичним робочим процесом. В результаті, під час співбесіди кандидати стикаються з упередженням ставленням, відсутністю стандартизації і питання про порівняння з зразком.

Пошук першого місця роботи - неабиякий стрес з яким стикається кожен. Для розробників співбесіда хвилююча, адже все знати неможливо, а отримувати неочікувані запитання, на які відповіді не знаєш, не дуже хороша практика.

Основною проблемою в відборі кандидата, є декілька рівнів співбесіди, на які кандидат і роботодавець витрачають багато часу. Тобто, спочатку технічний рівень, потім знання мови, потім дізнатись зарплатні очікування, та чим живе кандидат, тому що робота то тільки частина життя.

Оскільки, виникла така проблема, було вирішено створити додаток, який полегшить процедуру відбору, та зменшить час, який потрібно витрати до мінімуму. Це спростить життя як кандидатам, які не будуть хвилюватись і забувати все що знають під час співбесіди, і роботодавця, який буде точно знати, кого потрібно брати на роботу.

2.1.2 Функції системи

Основною функцією розроблюваної системи є алгоритм, який буде генерувати список питань, під конкретну професію, та рівень кандидата.

На головному екрані результату присутні такі речі:

- 1 Меню.
- 2 Статистика всіх опитувань відповідно до статусу.
- 3 Таблиця з результатами, та можливістю сортування.

Окрім головного екрану, існує низка інших необхідних екранів.

Екран реєстрації, який дає можливість авторизуватись, та переглядати інформацію користувачів.

Головний екран дає можливість переглянути статистику, та детальну інформацію по кожному опитуванні.

Меню додавання нового опитування, дозволяє додати опитування на конкретного користувача.

2.1.3 Джерело початкових даних

Джерелом початкових даних є база даних, яка повинна бути в системі, для того щоб можна було створити опитування.

Тобто, під час ініціалізації додатку, існує файл з розширенням «.csv», який містить бібліотеку питань, та відповідей до них. Дані питання завантажуються в базу даних, і потім ми можемо працювати з ними та здійснювати відповідні дії.

Додаток повинен зчитувати категорії питань та рівень, які має можливість вибрати користувач, і генерувати список, для конкретного працівника. Потім, відбувається генерація посилання. Сгенероване посилання пересилається працівнику. Працівник повинен перейти по даному посиланню, пройти тест, і потім очікувати результат. Лист питань, повинен мати унікальні питання і ділитись на 17 питань - тестові, 3 - розгорнута відповідь.

2.1.4 Результати, вихідні дані

Вихідними даними для додатку є результат тесту, який користувач може пройти тільки один раз. Після проходження, результат по кожному питанню,

записується в таблицю, яку ми можемо бачити на головній сторінці. Також, в цій таблиці є можливість сортувати та групувати результати кожного, хто пройшов опитування.

2.2 Вимоги до обчислювального середовища

Для надійного функціонування системи необхідно мати персональний комп'ютер, монітор, комп'ютерну мишу та клавіатуру. Для серверної частини, яка в свою чергу включає реєстрацію, зберігання даних та модуль обробки питань необхідний комп'ютер з наступними мінімальними вимогами:

- процесор з тактовою частотою: 1,5 ГГц;
- оперативна пам'ять: 2 Гб;
- відеокарта: дискретна або інтегрована;
- жорсткий диск: 1 Гб вільного простору;
- мережевий адаптер: 100 М/біт.

Для взаємодії з клієнтською частиною системи необхідна будь-яка ОС та сучасний браузер з підтримкою JavaScript.

Для сервера необхідним є Node.js версії 10.2.

2.3 Якість системи

Перелік вимог до якості за стандартом ISO 9126:2001:

- функціональність (functionality) — здатність ПЗ в певних умовах вирішувати задачі, потрібні користувачам. Визначає, що саме робить ПЗ, які задачі воно вирішує;
- функціональна придатність (suitability) — здатність вирішувати потрібний набір задач;
- точність (accuracy) — здатність видавати потрібні результати;
- здатність до взаємодії (interoperability) — здатність взаємодіяти з потрібним набором інших систем;
- відповідність стандартам і правилам (compliance) — відповідність ПЗ наявним індустріальним стандартам, нормативним і законодавчим актам, іншим регулюючим нормам;

- захищеність (security) — здатність запобігати неавторизованому, тобто без вказівки особи, що намагається його здійснити, і недозволеному доступу до даних і програм;
- надійність (reliability) — здатність ПЗ підтримувати визначену працездатність у заданих умовах;
- зрілість, завершеність (maturity) — величина, зворотна частоті відмов ПЗ. Звичайно вимірюється середнім часом роботи без збоїв і величиною, зворотною імовірності виникнення відмови за даний період часу;
- стійкість до відмов (fault tolerance) — здатність підтримувати заданий рівень працездатності при відмовах і порушеннях правил взаємодії з середовищем;
- здатність до відновлення (recoverability) — здатність відновлювати визначений рівень працездатності та цілісність даних після відмови, необхідні для цього час і ресурси;
- відповідність стандартам надійності (reliability compliance);
- зручність використання (usability) або практичність — здатність ПЗ бути зручним у навчанні та використанні, а також привабливим для користувачів;
- зрозумілість (understandability) — показник, зворотний до зусиль, які затрачаються користувачами на сприйняття основних понять ПЗ та усвідомлення їх застосовності для розв'язання своїх задач;
- зручність навчання (learnability) — показник, зворотний зусиллям, затрачуваним користувачами на навчання роботі з ПЗ;
- зручність роботи (operability) — показник, зворотний зусиллям, що вживається користувачами для розв'язання своїх задач за допомогою ПЗ;
- привабливість (attractiveness) — здатність ПЗ бути привабливим для користувачів;

- відповідність стандартам зручності використання (usability compliance);
- продуктивність (efficiency) або ефективність — здатність ПЗ при заданих умовах забезпечувати необхідну працездатність стосовно виділюваного для цього ресурсам. Можна визначити її і як відношення одержуваних за допомогою ПЗ результатів до затрачуваних на це ресурсів усіх типів;
- часова ефективність (time behaviour) — здатність ПЗ видавати очікувані результати, а також забезпечувати передачу необхідного об'єму даних за відведений час;
- ефективність використання ресурсів (resource utilisation) — здатність вирішувати потрібні задачі з використанням визначених об'ємів ресурсів визначених видів. Маються на увазі такі ресурси, як оперативна й довгострокова пам'ять, мережні з'єднання, пристрої вводу та виводу;
- відповідність стандартам продуктивності (efficiency compliance);
- зручність супроводу (maintainability) зручність проведення всіх видів діяльності, пов'язаних із супроводом програм;
- аналізованість (analyzability) або зручність проведення аналізу — зручність проведення аналізу помилок, дефектів і недоліків, а також зручність аналізу необхідності змін і їх можливих наслідків;
- зручність внесення змін (changeability) — показник, зворотний трудозатратам на виконання необхідних змін;
- стабільність (stability) — показник, зворотний ризику виникнення несподіваних ефектів при внесенні необхідних змін;
- зручність перевірки (testability) — показник, зворотний трудозатратам на проведення тестування і інших видів перевірки того, що внесені зміни привели до потрібних результатів;
- відповідність стандартам зручності супроводу (maintainability compliance);

- переносимість (portability) — здатність ПЗ зберігати працездатність при перенесенні з одного оточення в інше, включаючи організаційні, апаратні й програмні аспекти оточення;
- адаптованість (adaptability) — здатність ПЗ пристосовуватися різним оточенням без проведення для цього дій, крім заздалегідь передбачених;
- зручність установки (installability) — здатність ПЗ бути встановленим або розгорнутим у визначеному оточенні;
- здатність до співіснування (coexistence) — здатність ПЗ співіснувати з іншими програмами у загальному оточенні, ділячи з ними ресурси;
- зручність заміни (replaceability) іншого ПЗ даним — можливість застосування даного ПЗ замість інших програмних систем для вирішення тих же задач у певному оточенні;
- відповідність стандартам переносимості (portability compliance).

2.3.1 Виконання стандартів та узгодження

Якість розробленого модуля відповідає стандарту якості для програмного забезпечення ISO 9126:2001.

2.3.2 Можливість перенесення із точки зору апаратного та обчислюваного середовища

Система та її компоненти без жодних труднощів переносяться на інші комп'ютери, якщо комп'ютер відповідає програмним та апаратним вимогам, тому що система є кросплатформною та кросбраузерною (п. 2.3.1-2.3.2).

2.3.3 Надійність функціонування

Система буде надійно працювати якщо комп'ютер відповідає усім програмним та апаратним вимогам (п. 2.3.1-2.3.2).

2.4 Вимоги до оформлення вихідних кодів

Для коду написаного мовою програмування JavaScript, незалежно від версій, загальноприйнятими є певні правила, дотримуючись яких, можна полегшити життя тим людям, які будуть читати ваш код.

Існує декілька загальних правил. А саме:

- 1 Відступи, довжина рядка, обгортання коду, пробіли:
 - потрібно завжди використовувати 2 пробіли для відступу;
 - перед кожною дужкою повинен бути пробіл;
 - в об'єктах повинні бути пробіли;
 - пробіли в масивах;
 - як правило, використовуються саме пробіли, тому що вони дозволяють зробити більш гнучкі «конфігурації відступів», ніж символ «Tab»;
 - рівнів вкладеності має бути небагато. Наприклад, перевірки в циклах можна робити через «continue», щоб не було додаткового рівня if (..) {...};
 - краще швидко обробити прості випадки, повернути результат, а далі розбиратися зі складним, без додаткового рівня вкладеності;
 - як правило, краще розташовувати функції під кодом, який їх використовує;
 - довжина рядку повинна бути 80 символів;
 - 1 порожній рядок до функцій верхнього рівня та визначень класу;
 - ділення всього на функції, допомагає використовувати все як модулі, які ми можемо перевикористовувати;
 - для створення об'єкта використовуйте фігурні дужки. Не створюйте об'єкти через конструктор. Не потрібно робити декілька рівнів вкладеності, для створення об'єкта;
 - не використовуйте ключові слова (в тому числі змінні). Замість них використовуйте синоніми;
 - використовувати пробіли при математичних операціях;
 - відступ при вкладеності - 2 пробілу на кожен рівень вкладеності;
 - між логічними блоками (циклами, функціями і т.д.) слід залишати порожній рядок. Це робить код більш читабельним. Уникайте блоків коду більше 9 рядків поспіль;

- використовуйте пробіли між параметрами і не використовуйте між ім'ям функції і дужкою;
- при створенні анонімною функції необхідно використовувати пробіл перед дужкою;
- використовуйте пробіли навколо операторів;
- в результаті можна сказати, де слід відступити? Правило є простим - будь-де у фігурних дужках. Це означає тіла функцій, цикли (do, while, for, for, in, for-in), ifs, switches і властивості об'єкта в літературному позначенні об'єкта;
- Curly braces завжди слід використовувати, навіть у випадках, коли вони є необов'язковими. Технічно, якщо у вас є тільки одне твердження в чим-небудь, для фігурних фігурних фігурних дужок не потрібно, але ви повинні завжди використовувати їх у будь-якому випадку. Це робить код більш послідовним і простішим для оновлення;
- пробіли повинні бути після точок з крапкою, що відокремлюють частини циклу для циклу: наприклад, для (var i = 0; і Ініціалізація декількох змінних (i та max) у циклі для: для (var i = 0, max = 10; і Після коми, що обмежують елементи масиву: var a = [1, 2, 3]; після комами в властивостях об'єкта та після двокрапки, які розділяють імена властивостей та їх значення: var o = {a: 1, b: 2}; аргументи функції делімітації: myFunc (a, b, c) Перед фігурними фіксаціями в функціональних деклараціях: функція myFunc () {} Після функції в анонімних функціях: var myFunc = function () {}; Ще одним корисним використанням білого простору є відокремлення всіх операторів та їх операндів від пропусків, що в основному означає використання простору до і після +, -, *, =, ==, !=, &&, ||, += і т. д;
- в кінці кожного рядку, потрібно видаляти всі не потрібні пробіли, щоб не мати лишніх символів.

2 Коментарі:

- перевіряти актуальність коментарів - неправильні коментарі гірші, ніж без коментарів;
 - однорядкові коментарі починаються з подвійного слеша //. За ним обов'язково повинен йти пробіл;
 - багаторядкові коментарі розташовуються між / * і * /. За символом початку коментаря обов'язково повинен йти пробіл. Символ кінця коментаря розташовується на новому рядку;
 - повинен бути мінімум коментарів, які відповідають на питання 'що відбувається в коді?';
 - якщо вам здається, що потрібно додати коментар для поліпшення розуміння, це означає, що ваш код недостатньо простий, і, може, варто переписати його;
 - довідковий коментар перед функцією - про те, що саме вона робить, які параметри приймає і що повертає;
 - але куди більш важливими можуть бути коментарі, які пояснюють не що, а чому в коді відбувається саме це! Як правило, з коду можна зрозуміти, що він робить. Буває, звичайно, всяке, але, врешті-решт, ви цей код бачите. Однак набагато важливіше може бути те, чого ви не бачите!;
 - є кілька способів вирішення завдання. Чому обрано саме цей?;
 - які неочевидні можливості забезпечує цей код? Де ще вони використовуються?;
 - один з показників хорошого розробника - якість коментарів, які дозволяють ефективно підтримувати код, повертатися до нього після будь-якої паузи і легко вносити зміни;
 - використовуйте одиничні лапки всюди, крім випадків редагування JSON.
- 3 Умовні позначення:
- кожна нова змінна з новим словом var;

- використовувати стрілочні функції;
- оголошуйте по одній змінній з кожним ключовим словом `var`. Такий підхід полегшує переупорядочивання рядків. Не звертайте уваги на Crockford'a з цього приводу і розміщуйте оголошення там, де вони мають сенс;
- змінні і властивості повинні використовувати нижній верблюжий регістр. Крім того, вони повинні бути наочними. Одиначних символів і незвичайних аббревіатур слід уникати;
- імена класів повинні використовувати верхній верблюжий регістр;
- константи повинні оголошуватися як звичайні змінні або статичні властивості класів. Всі букви повинні бути в верхньому регістрі. Node.js / V8 підтримує розширення `const` від mozilla, але, на жаль, воно не може бути застосовано до членів класу, так само як і частина будь-якого стандарту ECMA;
- використовуйте завершальні коми і короткі оголошення на одному рядку. Ключі розміщуйте в лапки тільки тоді, коли інтерпретатор може не зрозуміти код;
- програмування - це вам не запам'ятовування правил. Використовуйте потрібний оператор рівності, так як тільки він буде працювати так, як очікується;
- не розширюйте прототипи всіх підряд об'єктів, особливо вбудованих. Це дуже погана практика. І навіть як що ви захочете таке зробити, шукайте спосіб щоб цей спосіб не використовувати;
- всі нетривіальні умови повинні присвоюватися змінній;
- функції повинні бути короткими. Гарна функція повинна вміститися на слайд, який буде добре читатися людьми, які сидять на останньому ряду в великій кімнаті;
- завжди повертайте значення з функції якомога раніше. Тим самим буде виключатися велика вкладеність `if-else` конструкцій;

- не бійтеся давати імена замиканням. Це показує, що ви дбаєте про них. Результатом цієї турботи буде більш повне трасування стека;
- не використовуйте вкладені замикання. Інакше ваш код перетвориться в кашу;
- так як Node.js насамперед зосереджений на неблокувальному введенні / виведенні, то функції в основному повертають свій результат, використовуючи callback'и. В ядрі Node.js прийнято угоду, що перший параметр будь-якого callback'а завжди є необов'язковим об'єктом-помилкою (error object);
- використовувати лінтер, він спрощує написання коду та слідкує за правильним написанням;
- не забувати про проблеми, які виникають під час спливання змінних;
- не використовуйте setter'и. У людей, які будуть намагатися використовувати ваш код, вони обов'язково викличуть масу проблем;
- Node.js включає в себе, серед іншого, простий клас EventEmitter, який може бути підключений з модуля 'events'. При створенні класів, які повинні підтримувати події, їх зазвичай успадковують від цього класу EventEmitter. В основному, це проста реалізація шаблону Observer. Однак, не рекомендується слухати події об'єкта зсередини самого об'єкта. Це не природно - слухати самого себе. Найчастіше, це веде до небажаного відкриття деталей реалізації та ускладнення супроводу коду;
- функція прив'язки є, ймовірно, найменшою вашою проблемою, коли ви починаєте з JavaScript, але коли ви розумієте, що вам потрібно вирішити проблему, як зберегти контекст цього в іншій функції, то ви можете зрозуміти, що вам насправді потрібна функція .prototype.bind ();
- закриття - це функції, які відносяться до незалежних (вільних) змінних. Іншими словами, функція, визначена в закритті,

«запам'ятовує» середовище, в якому вона була створена. Це важлива концепція для розуміння, оскільки вона може бути корисною під час розробки, наприклад, емуляція приватних методів. Це також може допомогти навчитися уникати типових помилок, як, наприклад, створення закриттів у петель;

- строгий режим ECMAScript 5 - це спосіб ввімкнути обмежений варіант JavaScript. Строгий режим - це не просто підмножина: вона навмисно має різну семантику від нормального коду. Браузери, які не підтримують суворий режим, виконуватимуть строгий код режиму з різною поведінкою від переглядачів, що роблять, тому не покладайтесь на строгий режим без тестування функцій для підтримки відповідних аспектів строгого режиму. Код суворого режиму і код суворого режиму можуть співіснувати, тому скрипти можуть вибирати в суворому режимі поступово;
- вираз функції, що безпосередньо викликається, є шаблоном, який створює лексичну область, що використовує функцію JavaScript. Негайно викликані вирази функцій можуть бути використані, щоб уникнути змінних підйомів зсередини блоків, захисту від забруднення глобального середовища і одночасно дозволити публічний доступ до методів, зберігаючи при цьому конфіденційність для змінних, визначених у функції;
- getter'и ж, навпаки, можна абсолютно спокійно використовувати в своєму коді. Так як вони є чистими функціями і вільні від побічних ефектів. Типовим прикладом є надання властивості довжини для класу колекції;
- розташовуйте відкриває дужку на тій же лінії, що і вираз.

2.5 Календарний план виконання робіт

Наведемо приклад календарно плану:

1	Підбір засобів розробки	26.02.19 - 11.03.19
2	Написання програмного коду	12.03.19 - 08.04.19

ЧДТУ.191547.012 ПЗ

3	Тестування	09.04.19 - 22.04.19
4	Доопрацювання	23.04.19 - 06.05.19

3 СТРУКТУРА ТА РОЗРОБКА

3.1 Розробка серверної частини

Серверна частина включає у себе декілька модулів :

- модуль реєстрації та авторизації;
- модуль бази даних;
- модуль обробки даних.

Вона написана на мові програмування Node.js версії 10.02. Основою серверної частини є фреймворк Express. Реєстрація та авторизація виконуються за допомогою бібліотеки Passport.js. Запити до бази даних та їх обробку виконує бібліотека Node-postgres. Ініціалізацією бази даних займається бібліотека Liquebase, вона створює класи, методи, міграції та наповнює базу.

Серверна частина містить такі компоненти: класи взаємодії з базою даних, класи взаємодії з front-end, що виконують обробку запитів з клієнту, конфігураційні файли, класи сутності (класи, що відображають сутності, які зберігаються у базі даних), клас налаштування liquebase.

3.1.1 Загальний опис алгоритму

Загальний алгоритм роботи складається з наступних кроків:

- 1 Виконується авторизація користувача через екран логіну.

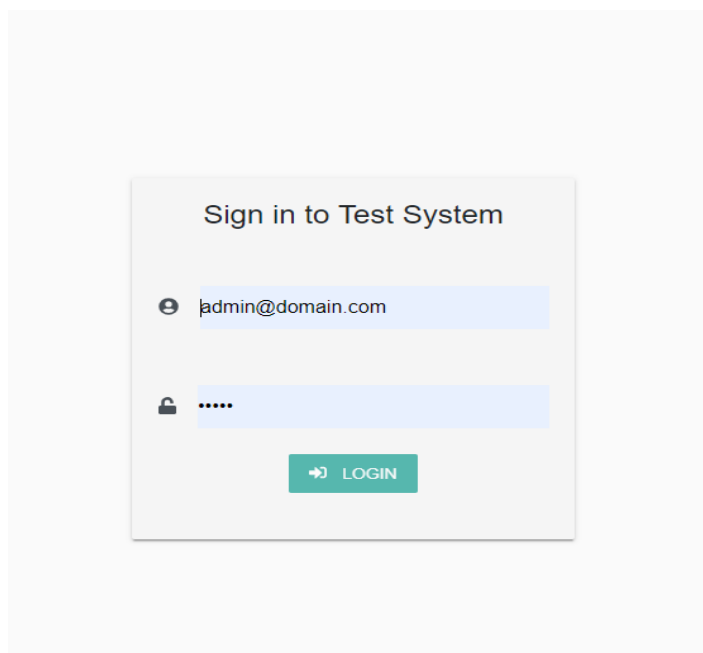


Рисунок 3.1 – Екран логіну

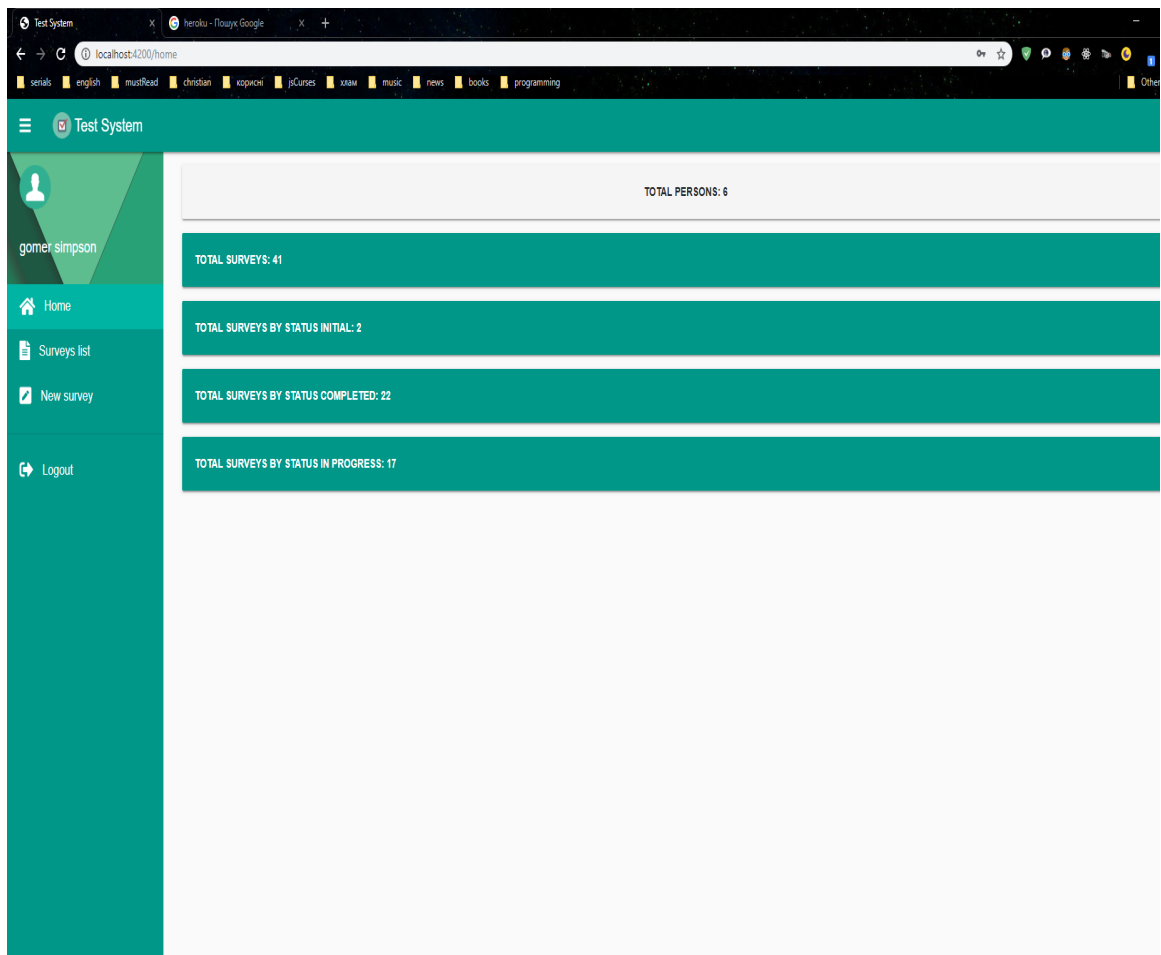


Рисунок 3.2 – Сторінка веб-додатку після авторизації

- 2 Виконується створення нового опитування. Ми повинні вибрати employee та його рівень. Потім натиснути кнопку Create survey.

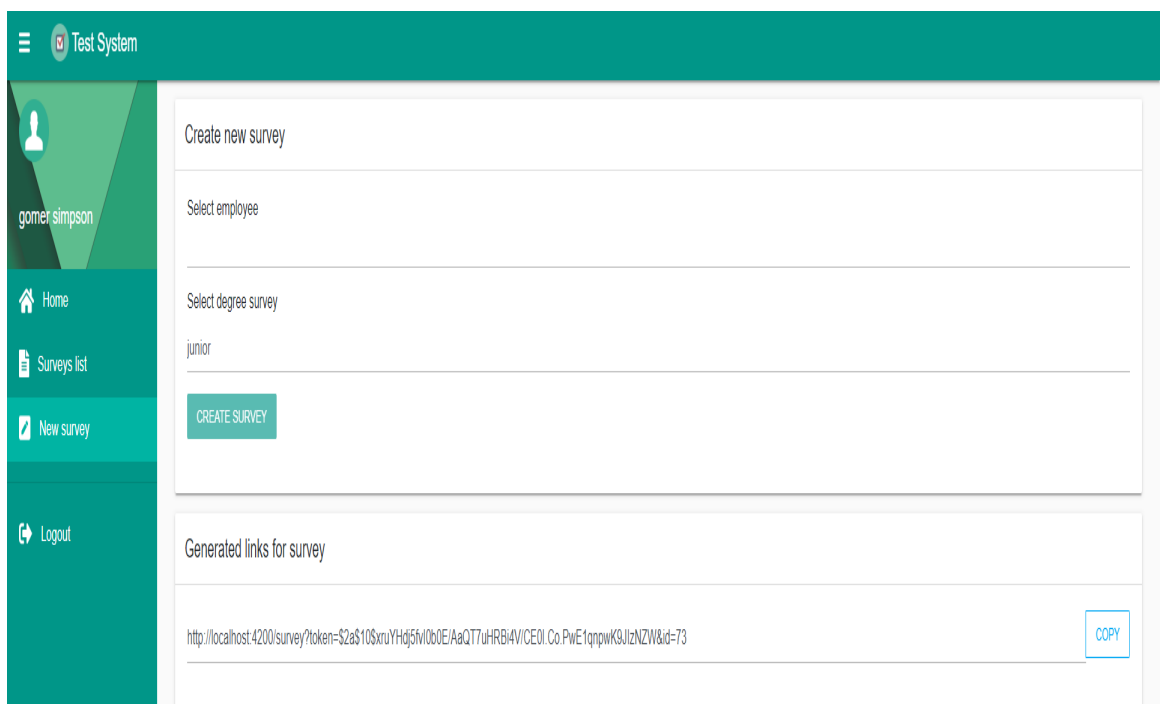


Рисунок 3.3 – Створення нового опитування

- 3 З екрану створення нового опитування слід скопіювати інформацію з “Generated links for survey”. І потім дану інформацію скопіювати в пошукову строку і нажати enter.

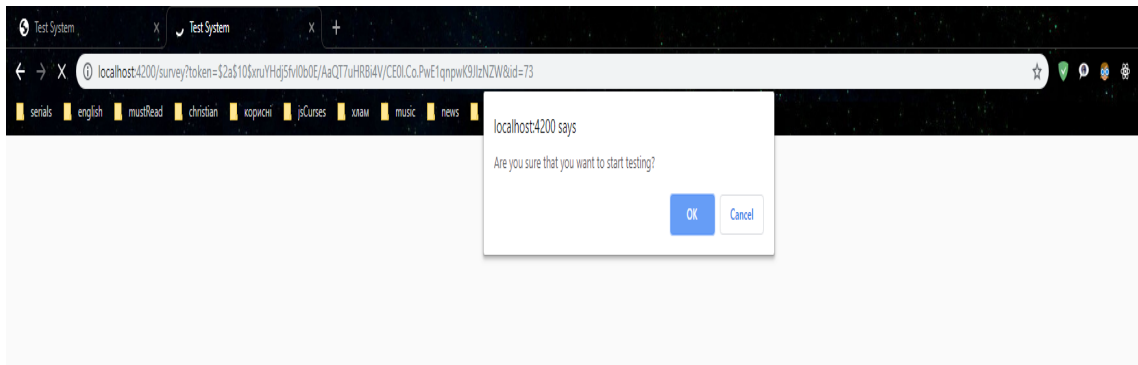


Рисунок 3.4 – Модальне вікно переходу на створене опитування

- 4 Після переходу на опитування, ви можете побачити модальне вікно. Після натиснення клавіші “Ok”, вам відкриється опитування, яке потрібно пройти.

Рисунок 3.5 – Проходження опитування

- 5 Після завершення тесту, ми відправляємо дані на сервер, і можемо бачити, що дані успішно відправлені.

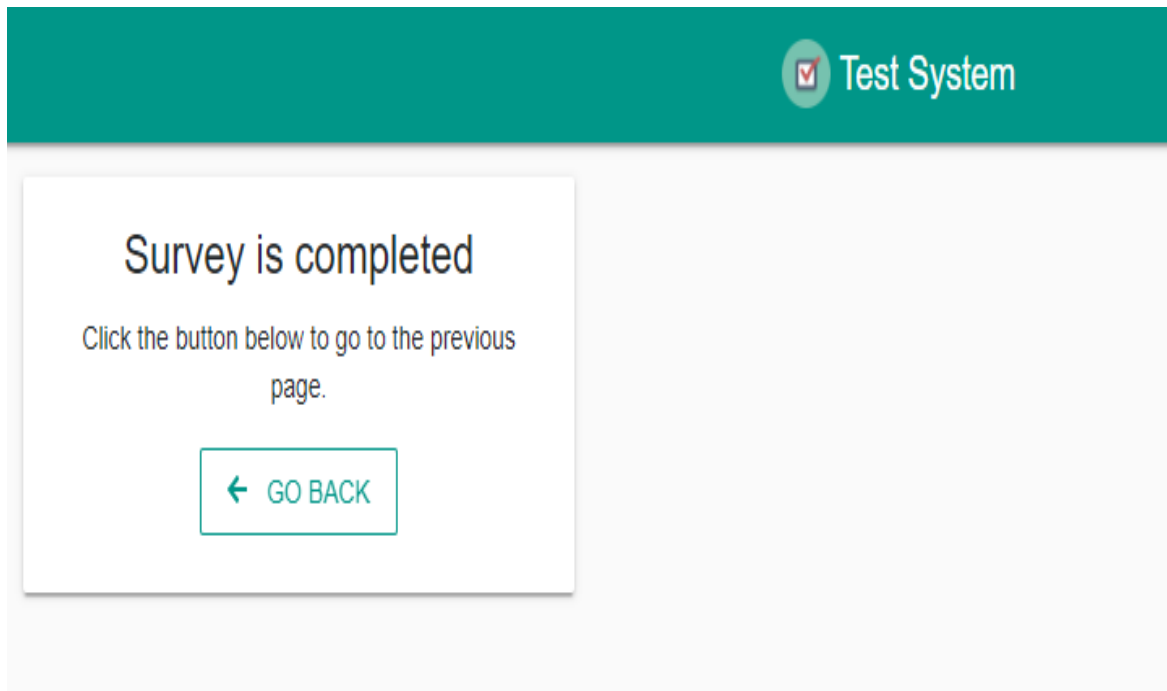


Рисунок 3.6 – Модальне вікно завершення опитування

- 6 Як тільки опитування було завершено, ми можемо подивитись статистику всіх пройдених опитувань.

The screenshot shows a web application header with a teal bar containing a logo and the text 'Test System'. Below the header, a sidebar on the left contains a user profile for 'gomer.simpson' and navigation links: 'Home', 'Surveys list', 'New survey', and 'Logout'. The main content area displays a table of survey results.

ID	First Name	Last Name	Status	Description	Show info
1	vlad	admin	in progress	1 junior	SHOW
35	Alexis	Ortiz	completed	5 senior	SHOW
68	joker	.	completed	8 middle-1	SHOW
69	joker	.	in progress	8 junior	SHOW
2	vlad	admin	completed	1 middle-1	SHOW
3	vitali	not admin	in progress	2 junior	SHOW
4	vitali	not admin	in progress	2 middle-1	SHOW
5	vitali	not admin	in progress	2 junior	SHOW
6	vitali	not admin	in progress	2 junior	SHOW

Рисунок 3.7 – Статистика результатів опитування

- 7 При переході на кнопку “Show info”, відкривається детальна інформація по опитуванню.

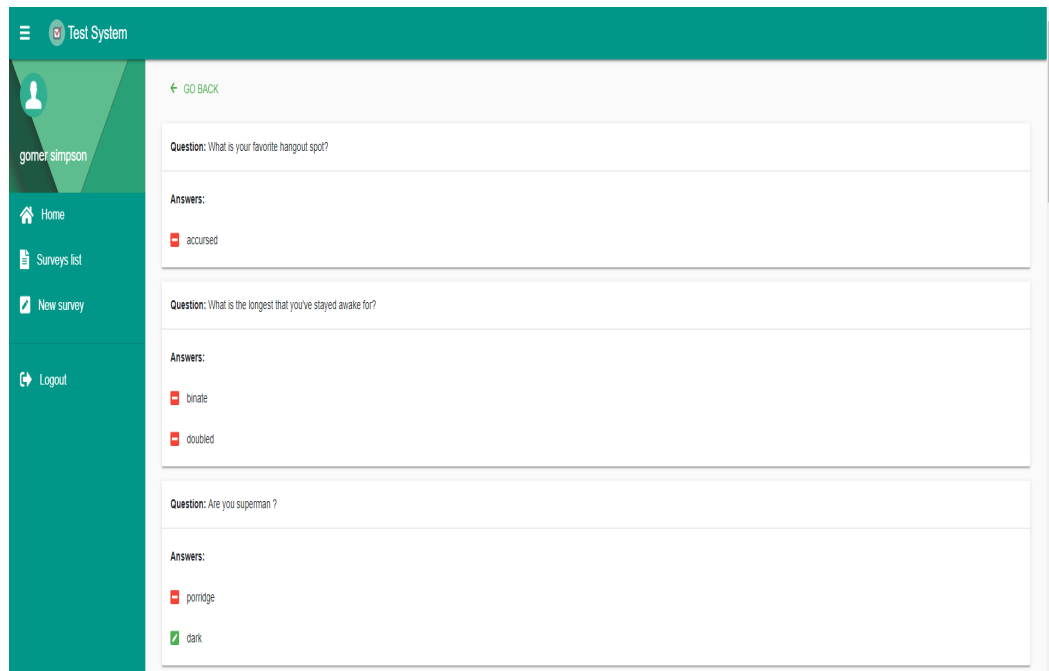


Рисунок 3.8 – Детальна інформація по опитуванню

3.1.2 Розробка структури бази даних

Розробка бази даних викликана необхідністю збереження вхідних та вихідних даних. Також, база даних повинна мати міграції, для того щоб додаток працював без помилок. Тому була розроблена база даних яка б відповідала вищезазначеним функціональним особливостям. В якості бази даних обрано PostgreSQL, переваги цієї системи вказані у п. 1.3. У розробленій системі можна відокремити такі сутності:

- 1 Person — користувач системи, використовується для керування правами доступу та авторизації.









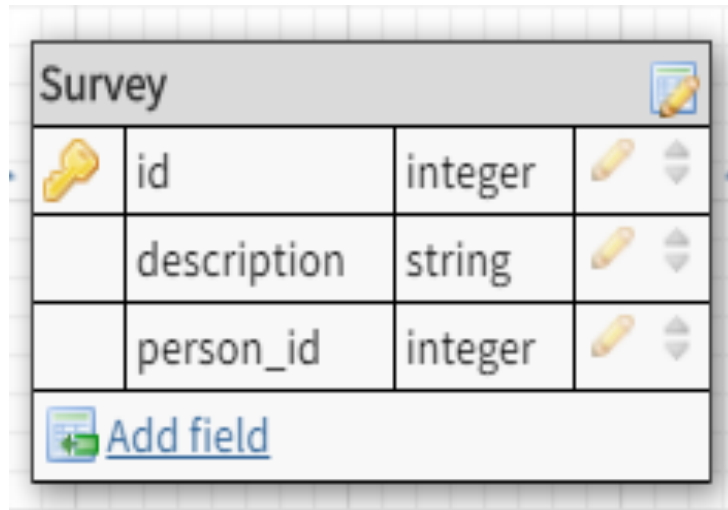
Person			
	id	integer	
	email	string	
	is_admin	boolean	
	hash	string	
	firstname	string	
	lastname	string	
 Add field			

Рисунок 3.9 – Структура таблиці Person

Має відповідні поля, та права доступу відповідно до флагу `is_admin`. Пароль хешується, та в таблицю записується хеш. Потім, під час входу, перевіряється хеш. Цей спосіб зберігання даних підвищує безпеку для користувачів.

- 2 Survey – сутність, яка зберігає опитування та до якого користувача воно відноситься. Такий спосіб опитування допомагає зберегти гнучкість системи.








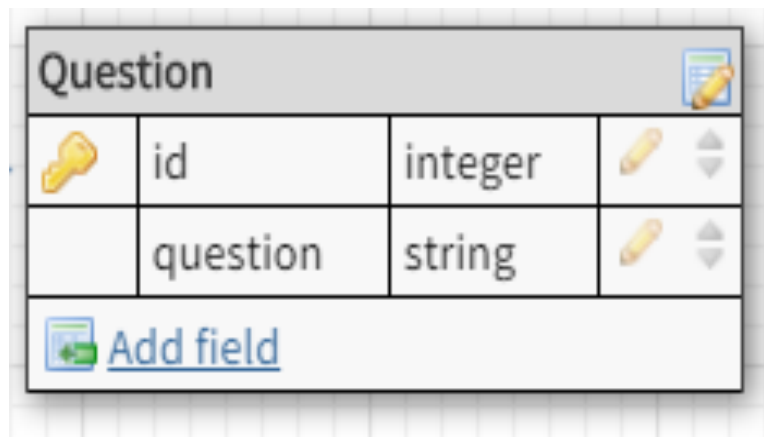
Survey			
	id	integer	
	description	string	
	person_id	integer	
 Add field			

Рисунок 3.10– Структура таблиці Survey

- 3 Question – сутність яка містить назву питання. Також, ми можемо зберігати певний опис питання, якщо потрібно.







Question			
	id	integer	
	question	string	
 Add field			

Рисунок 3.11 – Структура таблиці Question

- 4 Survey_Questions – проміжна сутність, яка поєднує опитування із питаннями. В таблиці зберігається id опитування та питання до нього. Цей зв'язок має назву багато – до одного. Тобто, одне опитування, багато питань.






Survey_Questions			
	id	integer	
	survey_id	integer	
	question_id	integer	
 Add field			

Рисунок 3.12 – Структура таблиці Survey_Questions

- 5 Question_Answers – проміжна сутність, яка зєднує питання і відповіді.







Question_Answers			
	id	integer	
	question_id	integer	
	answer	text	
	is_right	boolean	
 Add field			

Рисунок 3.13– Структура таблиці Question_Answers

- 6 Question_Person_Answers – проміжна сутність, яка поєднує відповіді певного користувача, з опитуванням.







Question_Person_Answers			
	id	integer	
	survey_id	integer	
	question_answers_id	integer	
	full_answer	text	
 Add field			

Рисунок 3.14– Структура таблиці Question_Person_Answers

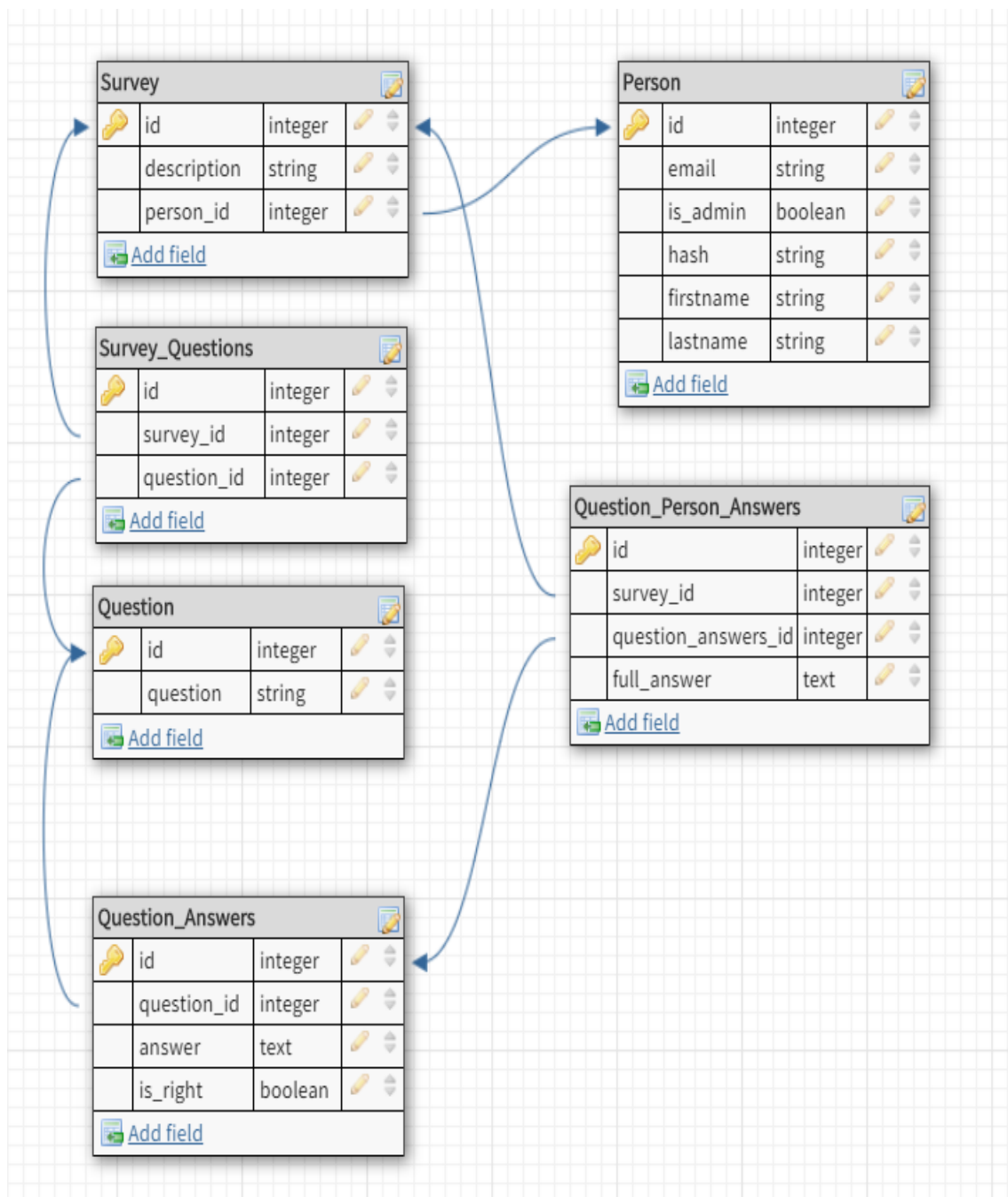


Рисунок 3.15 – UML діаграма бази даних

3.1.3 Опис файлів з вихідним кодом

У цьому розділі описуються файли програми, їх функціональне призначення, а також особливості обраних архітектурних рішень. Повний код цих файлів додається у додатку Б.

index.js – файл, що здійснює створення серверу, та його налаштування. Також містить ініціалізацію пакетів.

.gitignore – файл, що містить список тих файлів в репозиторії, які повинні ігноруватись системою версій git.

app.js – файл, що містить налаштування для ініціалізації проекту на платформі Heroku.

profile – файл, що містить налаштування для ініціалізації проекту на платформі Heroku.

readmy.md – файл, що містить опис репозиторію, та всіх файлів що в ньому описано. Інформація показується при відкриванні аккаунту.

connection.js – файл, що здійснює підключення до PostgreSQL бази даних, за допомогою налаштувань.

liquebaseSettings.js – файл, що містить авторизацію та доступ бібліотеки liquebase до БД.

Licence – файл, що містить ліцензію на використання пакету liquebase.

liquebase.js – файл, що містить налаштування для бібліотеки, та її доступ до конфігураційних налаштувань.

passportLocal.js – конфігураційний файл, що містить класи, які відповідають за реєстрацію та авторизацію користувача. Також, має налаштування для роботи з бібліотекою passport.js. Створює унікальний токен, для верифікації кожного користувача в системі.

authRoutes.js – файл, що містить шляхи, для додавання, редагування та зміни інформації, в базі даних.

degreeRoutes.js – файл, що містить шляхи, для додавання, редагування та зміни інформації, в базі даних, що відноситься до таблиці Degree.

personRoutes.js – файл, що містить шляхи, для додавання, редагування та зміни інформації, в базі даних, що відноситься до таблиці Person.

questionPersonAnswerRoutes.js – клас, що виконує обробку HTTP запитів пов'язаних з CRUD операціями над об'єктами типу Photo.

routes.js – клас, що виконує обробку HTTP запиту для збереження проаналізованих фото у об'єктів типу Result у базу даних.

surveyQuestionRoutes.js – клас, що надає доступ до реєстрації користувачів шляхом виклику POST методу.

surveysRoutes.js – клас, що взаємодіє з таблицею surveys, та налаштовує шляхи, для доступу з сторони клієнту.

package.json – список пакетів, які потрібні для корекної роботи додатку, та їхні версії.

ВИСНОВКИ

В ході виконання даної бакалаврської роботи був розроблений web-додаток «Система тестування рівня кваліфікації працівника». Результати розробки наведені у додатку А. Додаток дозволяє спростити процес як прийому на роботу, так і підвищення рівня кваліфікації працівника.

В процесі розробки системи, проект пройшов всі необхідні етапи, а саме: огляд предметної області, проектування, реалізація, тестування та впровадження.

На етапі огляду предметної області було досліджено аналоги, та виявлено їх сильні та слабкі сторони.

На етапі проектування, було спроектовано діаграму бази даних, яка включає всі потреби для реалізації додатку.

На етапі реалізації, були реалізовані модулі: реєстрація, авторизація, імпорт даних з файлів, створення опитування, та статистика.

На етапі тестування, було протестовано якість маршрутів, за допомогою програми Postman. Також, тестувався код, та виконувався рефакторинг. Були проведені тести швидкості запитів, та їх апгрейд.

На етапі впровадження, модулі системи були викладені на Heroku.

Для розробки системи були використані такі програмні засоби, як: WebStorm, Git, postgresql, Heroku. Для описання алгоритмів роботи системи було використано мову програмування JavaScript.

Всі елементи системи були розміщені в процесі розробки в системі контролю версій Github.

Розроблена система готова до її використання в постійному режимі користувачами. Вона має зручний та інтуїтивно зрозумілий інтерфейс. Також, система може бути розширена додатковими функціональними можливостями, такими як можливість вибору мови програмування, індивідуальний кабінет для користувача та адміна, зміна мови.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1 Офіційний сайт Node [Електронний ресурс]:[Інтернет-портал]. – Електронні дані. – Режим доступу: <https://nodejs.org/en> (дата звернення 01.03.19). – Node.
- 2 Сайт модулів для Node.js [Електронний ресурс]:[Інтернет-портал]. – Електронні дані. – Режим доступу: <https://www.npmjs.com> (дата звернення 05.03.19). – npm.
- 3 Офіційний сайт Liquibase [Електронний ресурс]:[Інтернет-портал]. – Електронні дані. – Режим доступу: <https://www.liquibase.org> (дата звернення 05.03.19). – Liquibase.
- 4 Liquibase налаштування [Електронний ресурс]:[Інтернет-портал]. – Електронні дані. – Режим доступу: <https://habr.com/ru/post/436994> (дата звернення 01.04.19). – Liquibase.
- 5 Офіційний сайт Nodemailer [Електронний ресурс]:[Інтернет-портал]. – Електронні дані. – Режим доступу: <https://nodemailer.com/about> (дата звернення 06.04.19). – nodeMailer.
- 6 Офіційний сайт Postgresql [Електронний ресурс]:[Інтернет-портал]. – Електронні дані. – Режим доступу: <https://www.postgresql.org/docs/9.6/plpgsql.html> (дата звернення 15.04.19). – postgresql.
- 7 Офіційний сайт Typescriptlang [Електронний ресурс]:[Інтернет-портал]. – Електронні дані. – Режим доступу: <https://www.typescriptlang.org>(дата звернення 15.05.19). – JavaScript.
- 8 Вивчення javascript [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: <https://learn.javascript.ru> (дата звернення 18.05.19). – JavaScript.
- 9 Бібліотека для роботи з Postgresql на node.js [Електронний ресурс]:[Інтернет-портал]. – Електронні дані. – Режим доступу: <https://www.npmjs.com/package/pg> (дата звернення 20.05.19). – postgresql.

10 Офіційний сайт Heroku [Електронний ресурс]:[Інтернет-портал]. – Електронні дані. – Режим доступу: <https://www.heroku.com> (дата звернення 21.05.19). – heroku.

11 Ініціалізація node.js на heroku [Електронний ресурс]:[Інтернет-портал]. – Електронні дані. – Режим доступу: <https://appdividend.com/2018/04/14/how-to-deploy-nodejs-app-to-heroku> (дата звернення 21.05.19). – heroku.

12 Async / Await в Node.js та їх використання в додатку для node.js [Електронний ресурс]:[Інтернет-портал]. – Електронні дані. – Режим доступу: <https://habr.com/ru/post/334772/> (дата звернення 21.05.19). – JavaScript.

ЧДТУ.191547.012 ПЗ

ДОДАТКИ