МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК, ФІЗИКИ ТА МАТЕМАТИКИ
КАФЕДРА ІНФОРМАТИКИ, ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ЕКОНОМІЧНОЇ КІБЕРНЕТИКИ

РОЗРОБЛЕННЯ WEB-ДОДАТКУ ДЛЯ ЗДІЙСНЕННЯ ПЕДАГОГІЧНОГО ТЕСТУВАННЯ

Кваліфікаційна робота (проєкт)

на здобуття ступеня вищої освіти «бакалавр»

Виконав: студент 4 курсу
Спеціальності 121 Інженерія програмного забезпечення
Освітньо-професійної програми
«Інженерія програмного забезпечення»
першого (бакалаврського) рівня освіти
Шувалов Андрій Кирилович
Керівник доктор педагогічних наук,
професор Шерман Михайло Ісаакович
доктор педагогічних наук,
професор Співаковський Олександр Володимирович
Рецензент кандидат педагогічних наук,
старший викладач
Григор'єва Валентина Борисівна

Херсон – 2020

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД	База даних
ПК	Персональний комп’ютер
AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
CRUD	Create Read Update Delete
DOM	Document Object Model
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
JSON	JavaScript Object Notation
MVC	Model View Controller
MVVM	Model View ViewModel
NoSql	Not only Structured Query Language
SDK	software development kit
SQL	Structured Query Language
SSL	Secure Sockets Layer
URL	Uniform Resource Locator

ВСТУП

На сьогоднішній день, багато процесів та явищ в нашому житті переходять у інформаційний простір. Люди спілкуються в соціальних мережах, працюють за комп'ютерами та проводять вільний час за переглядом серіалів або відеоіграми. Сьогодні мабуть вже в кожного є смартфон або ноутбук. Саме тому діджіталізація всіх сфер людської діяльності є лише питанням часу. Не є виключенням і освіта. В нашій державі, а також в інших країнах колишнього радянського союзу весь освітній процес вже перейшов інформаційний простір. І ця тенденція є зрозумілою, адже набагато зручніше та надійніше зосередити весь освітній процес у своєму смартфоні чи ПК.

З огляду на це, контроль за рівнем знань, а також опитування на ті чи інші теми, що стосуються процесу навчання також набагато доцільніше перевести у інформаційний простір. Тому я вирішив розробити інструмент у вигляді веб-додатку, який зможе надати студентам та викладачам можливість швидко проходити різного роду опитування.

Актуальність теми. Інструмент для проходження опитувань та тестувань для всіх учасників навчального процесу.

Мета дослідження – розробка веб-додатку для здійснення педагогічного тестування.

Завдання дослідження:

1. Аналіз аналогів.
2. Визначення вимог до веб-додатку.
3. Складання діаграм прецендентів .
4. Розглядання сучасних підходів та технологій розробки мобільних додатків.

5. Розроблення веб-додатку.

Об’єкт дослідження: сучасні технології створення веб-додатків.

Предмет дослідження: веб-додаток для проходження опитувань.

Практичне значення полягає в наданні студентам та викладачам вищих навчальних закладів можливість проходження опитувань в гарному та зручному вигляді.

Структура дослідження. Дипломна робота складається зі вступу, списку умовних скорочень, трьох розділів, висновків, списку використаних джерел.

РОЗДІЛ 1.
АНАЛІЗ АНАЛОГІВ ТА ВИЗНАЧЕННЯ ВИМОГ

1.1. Постановка задачі

Створення будь-якого продукту починається з постановки задачі. В нашому випадку задача – це розробити веб-додаток для проходження педагогічного тестування, який буде володіти усіма потрібними якостями та задовольняти потреби користувачів.

На сьогоднішній день майже у всіх студентів є смартфон з виходом в інтернет, тому й проходити опитування їм буде набагато зручніше за допомогою смартфону. Також в додатку можна буде зберігати створене опитування в «Чернетки» та потім, коли знадобиться, запускати його в пару кліків. І, можливо, викладачам це теж буде зручніше робити зі смартфону. Тому при розробленні треба враховувати можливість користування додатком як з ПК, так і зі смартфону.

Також всі тестування повинні зберігатись, для перегляду викладачем. Окрім цього викладач повинен мати змогу легко створювати копії типових тестувань зі зміною потрібних йому параметрів. Це передбачає наявність серверної частини та БД.

Постановка задачі: розробити веб-додаток для проходження опитувань та тестувань. Звести функціонал до мінімуму, обмежившись цільовими запитами користувачів. Ознайомитись з аналогами та сучасними технологіями розроблення веб-додатків. Визначити вимоги до веб-додатку. Забезпечити належний рівень швидкодії веб-додатку та доцільну роботу серверної частини.

1.2. Аналіз аналогів

Розроблення мобільних додатків – це складний технологічний процес, що вимагає ретельного планування та виконання. Перед початком написання вимог продукту, проєктуванням та процесом розробки необхідно провести аналіз аналогів. Наш продукт не є виключенням. Для проведення аналізу були обрані веб-додатки «Анкетолог» (Рис. 1.1.) та Survio (Рис. 1.2.). Після аналізу було виявлені основні недоліки та особливості, на які було звернено увагу:

«Анкетолог» (https://anketolog.ru/)

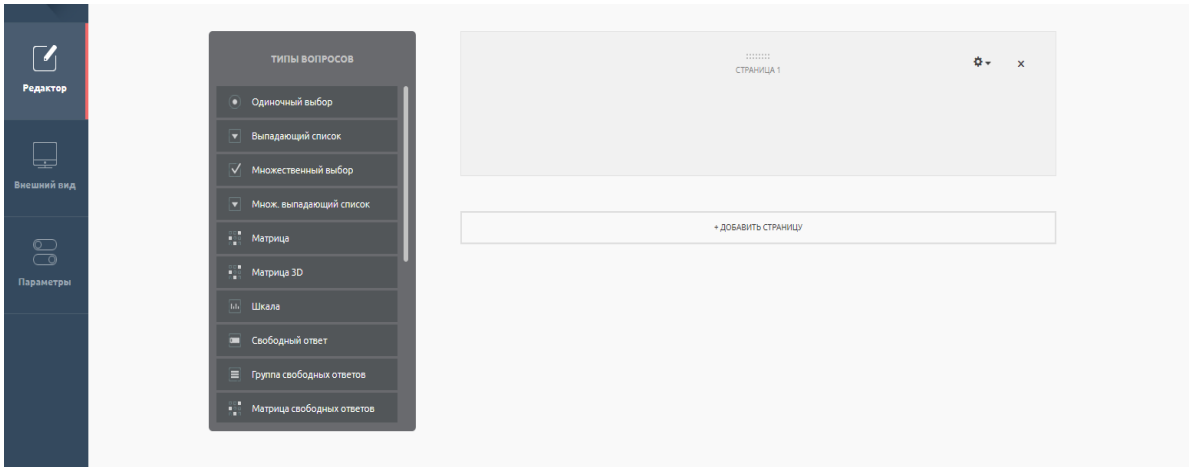

Рис. 1.1. «Анкетолог»

Переваги:

1. 14 типів питань та налаштування логічних правил.
2. Онлайн-панель з 13200 активних респондентів, та 9 засобів збору відповідей.
3. Результати опитування у форматі *.pdf, *.docx, *.xlsx. Дані у вигляду графіків та діаграм.

Недоліки:

1. Платні тарифи на користування веб-додатком
2. Функціонал великий, але користування незручне

«Survio» (https://my.survio.com)

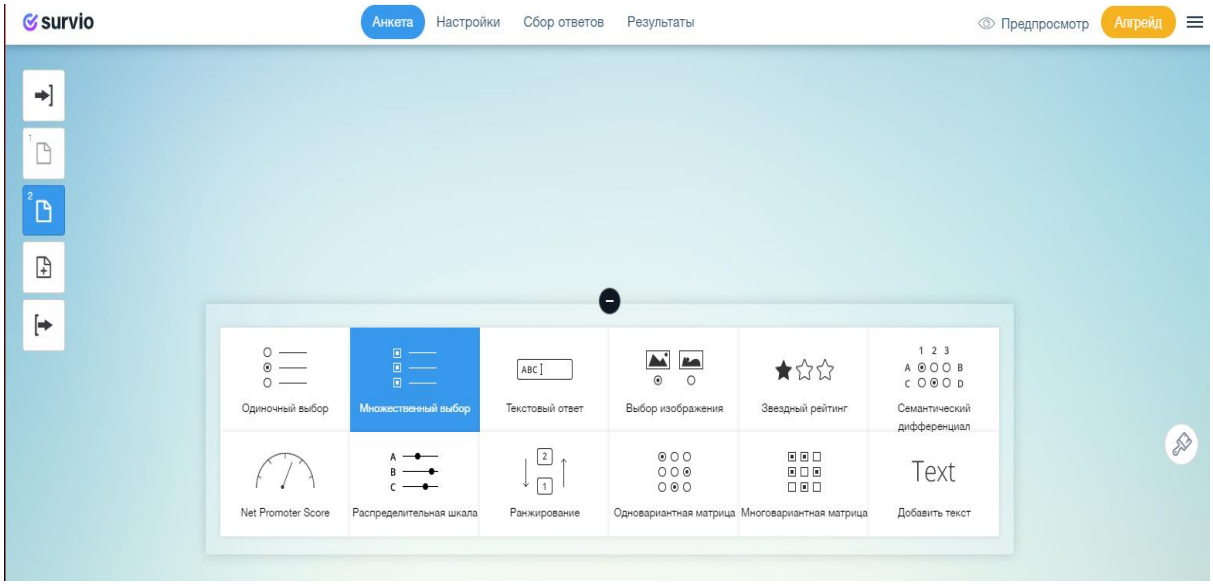

Рис. 1.2. «Survio»

Переваги «Survio»**:**

1. Мінімалізм у функціоналі та дизайну. Це зручно з точки зору користувача.
2. Є можливість використати QR-код для поширення анкети
3. Не поганий функціонал налаштувань
4. Щоденний back-up

Недоліки «Survio»**:**

1. Інде не працюючий функціонал
2. Дизайн не відповідає сучасності
3. Функціонал недостатньо об’ємний

З точки зору розробки ці додатки є недопрацьованими, адже не на всіх екранах дотримано адаптивності усіх елементів, та в деяких моментах спілкування з серверною частиною ми можемо спостерігати дуже великий за часом відгук.

1.3. Загальні вимоги до додатку

Вимоги до програмного забезпечення - сукупність тверджень щодо атрибутів, властивостей або якостей програмної системи, яка підлягає реалізації.

На основі поставлених задача та проведеного аналізу аналогів був визначений ряд головних вимог до продукту:

- Коректне відображення на екранах усіх розмірів;
- Мова інтерфейсу – українська;
- Належний рівень швидкодії;
- Збереження створених та пройдених опитувань в оптимальному вигляді;
- Гнучка архітектура для можливості подальшого масштабування;
- Достатньо низький рівень відгуку у всіх моментах спілкування клієнту з сервером.

1.4. Створення діаграми прецендентів

Діаграми прецендентів застосовуються для моделювання виду системи з точки зору прецедентів (або варіантів використання). Найчастіше це передбачає моделювання контексту системи, підсистеми або класу або моделювання вимог, що пред'являються до поведінки зазначених елементів.

На діаграмі прецендентів можна побачити, що користувач спочатку авторизується, і в залежності від введених їм даних потрапляє до екрану опитування або до адмін-панелі.

Опитуваному відкривається екран опитування по введеному ключу. Після проходження опитування, результати відправляються на сервер, а опитуваний потрапляє на сторінку авторизації.

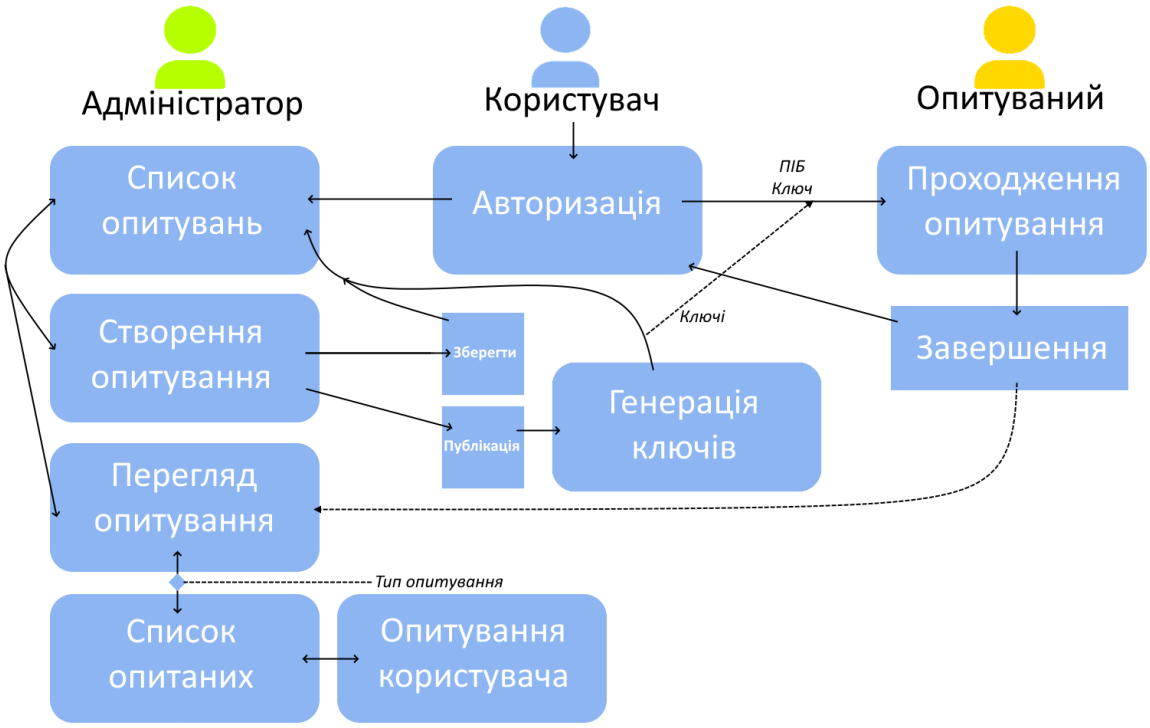

Рис. 1.3. Діаграма прецендентів

Адміністратору відкривається екран зі списком опитувань, які він може переглядати. В деяких опитуваннях (в залежності від типу) він може переглядати відповіді конкретних користувачів вибравши потрібного зі списку за ключем та ПІБ. Також адміністратор може створювати нові опитування, зберігати їх та публікувати, отримуючи ключі для проходження.

РОЗДІЛ 2.
СУЧАСНІ ПІДХОДИ ТА ТЕХНОЛОГІЇ РОЗРОБЛЕННЯ

2.1. Архітектура клієнт-сервер

В основу сервісу закладена дворівнева архітектура клієнт-серверної взаємодії. Принцип роботи дворівневої архітектури взаємодії клієнт-сервер полягає в тому, що обробка запиту відбувається на одній машині (сервері) без використання сторонніх ресурсів.

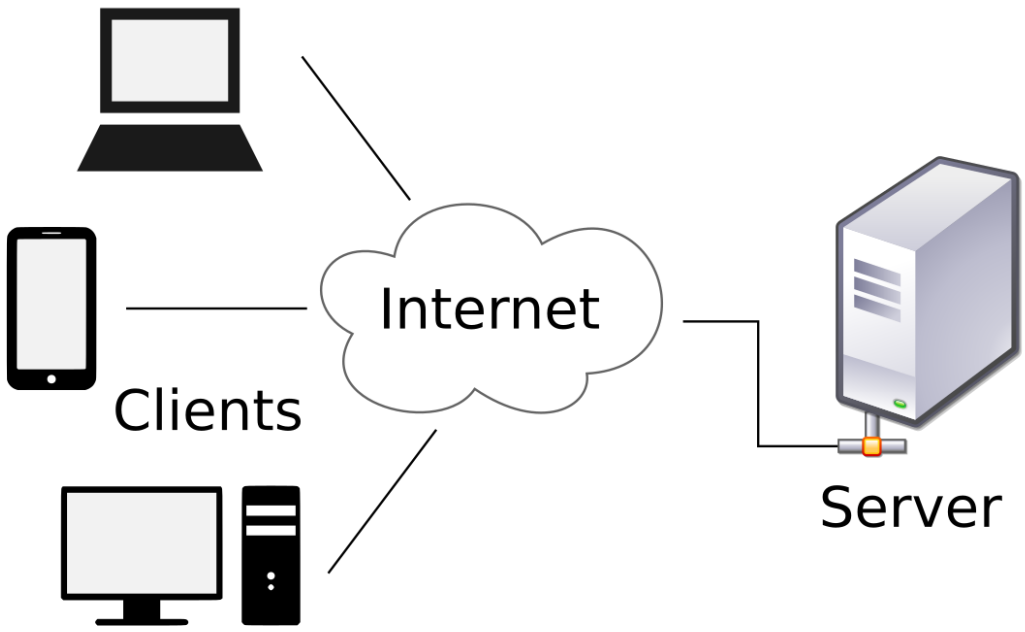

Рис. 2.1 Клієнт-серверна архітектура

Клієнт і сервер взаємодію один з одним в мережі Інтернет або в будь-який інший комп'ютерної мережі за допомогою різних мережевих протоколів, наприклад, IP протокол, HTTP протокол, FTP та інші. Протоколів насправді дуже багато і кожен протокол дозволяє надавати ту чи іншу послугу. В нашому сервісі взаємодія між клієнтом та сервером відбувається завдяки HTTP.

Взаємодія клієнта і сервера завжди починається саме з клієнта. Сервер ж вирішує, чи давати відповідь клієнту, і якщо так - то яку саме. На стороні клієнта же йде обробка отриманих даних, і відображення їх користувачу. За рахунок того, що всі важкі операції, такі як взаємодія з базою даних, виконуються на стороні сервера, ми істотно знижуємо вимога до машин клієнтів.

Платформою для реалізації клієнтської частини в нашому випаду виступає веб-додаток.

2.2. Принципи роботи веб-додатків

Веб-додаток являє собою веб-сайт, на якому розміщені сторінки з частково або повністю несформованим вмістом. Остаточний вміст формується тільки після того, як відвідувач сайту запросить сторінку з веб-сервера. У зв'язку з тим що остаточне вміст сторінки залежить від запиту, створеного на основі дій користувача, така сторінка називається динамічною.

Будь-який веб-додаток являє собою набір статичних і динамічних веб-сторінок. Статична веб-сторінка – це сторінка, яка завжди відображається перед користувачем в незмінному вигляді. Веб-сервер відправляє сторінку за запитом веб-браузера без будь-яких змін. На противагу цьому, сервер вносить зміни в динамічну веб-сторінку перед відправкою її браузеру. У зв'язку з тим що сторінка змінюється, вона називається динамічною.

Для прикладу, можна взяти сторінку, на якій будуть відображені результати опитування. При цьому деяка інформація (наприклад, ім'я співробітника і його результати) буде визначатися в момент запиту сторінки співробітником.

Статичний веб-сайт містить набір відповідних HTML-сторінок і файлів, розміщених на комп'ютері, на якому встановлено веб-сервер.

Веб-сервер - це програмне забезпечення, яке надає веб-сторінки у відповідь на запити веб-браузерів. Зазвичай запит сторінки створюється при натисканні посилання на веб-сторінці, виборі закладки в браузері або введенні URL-адреси сторінки в адресному рядку браузера.

Остаточний вміст статичної веб-сторінки визначається розробником і залишається незмінним у процесі запиту сторінки. Весь HTML-код створюється розробником до того моменту, коли сторінка буде розміщена на сервері. Оскільки HTML-код не змінюється після розміщення сторінки на сервері, дана сторінка називається статичною.

Коли веб-сервер отримує запит на видачу статичної сторінки, то, після аналізу запиту, сервер знаходить потрібну сторінку і відправляє її браузеру, (Рисунок ниже)

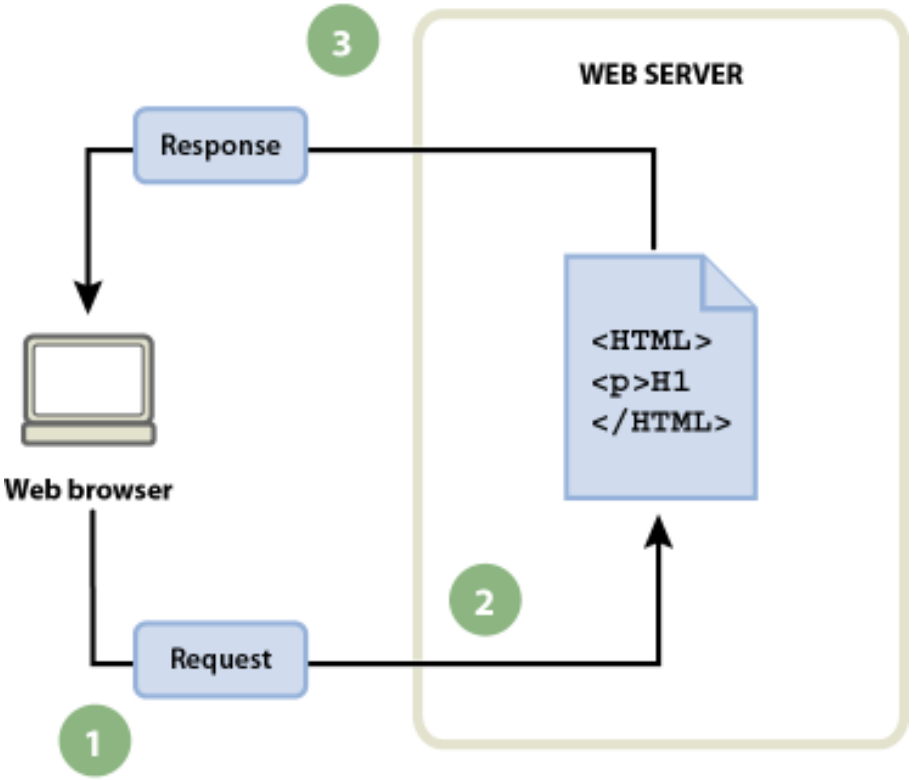

Рис. 2.2.1. Запит статичної сторінки

У випадку веб-додатків деякі ділянки коду сторінки відсутні до моменту запиту сторінки відвідувачем. Відсутній код формується за допомогою деякого механізму, і тільки після цього сторінка може бути відправлена браузеру.

Коли веб-сервер отримує запит на видачу статичної веб-сторінки, він відправляє сторінку безпосередньо браузеру. Однак, коли запитується динамічна сторінка, дії веб-сервера не настільки однозначні. Сервер передає сторінку спеціальною програмою, яка і формує остаточну сторінку. Така програма називається сервером додатків.

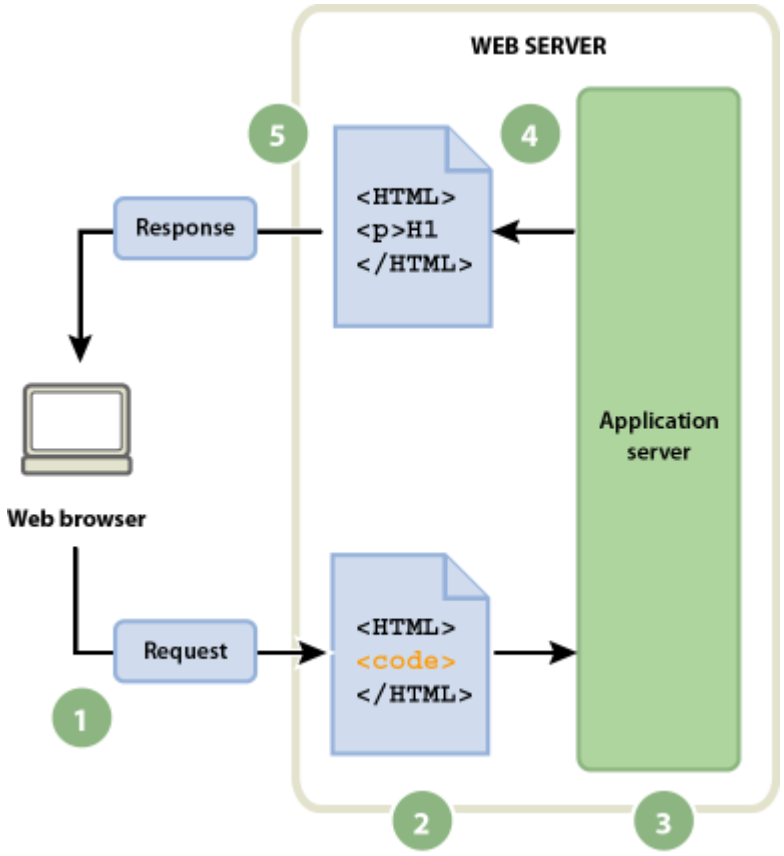

Рис. 2.2.2. Запит статичної сторінки

Сервер додатків виконує читання коду, що знаходиться на сторінці, формує остаточну сторінку відповідно до прочитаним кодом, а потім видаляє його з сторінки. В результаті всіх цих операцій виходить статична сторінка, яка передається веб-сервера, який в свою чергу

відправляє її клієнтському браузеру. Всі сторінки, які отримує браузер, містять тільки HTML-код.

Зберігання вмісту в базі даних дозволяє відокремити оформлення веб-сайту від вмісту, яке будуть бачити користувачі. Замість того щоб створювати всі сторінки у вигляді окремих HTML-файлів, пишуться тільки шаблони сторінок для кожного виду інформації, що представляється. Потім вміст завантажується в базу даних, після чого веб-сайт буде витягувати його при запитах користувачів. Крім того, можна оновити інформацію в одному джерелі і продублювати це зміна на всіх веб-сайті без редагування кожної сторінки вручну.

Програмна інструкція, призначена для отримання даних з бази даних, називається запитом до бази даних. Запит складається з критеріїв пошуку, виражених за допомогою мови баз даних, званого SQL (мова структурованих запитів). Текст SQL-запиту розташовується в сценаріях сторінок на стороні сервера або в тегах.

Сервер додатків не може безпосередньо отримати дані з бази, оскільки бази даних використовують специфічні формати зберігання даних, в результаті чого спроба отримання таких даних буде нагадувати спробу відкриття документа Microsoft Word за допомогою текстового редактора Notepad або BBEdit. Тому для підключення до бази даних сервер додатків використовує посередника - драйвер бази даних. Драйвер бази даних являє собою програмний модуль, за допомогою якого встановлюється взаємодія між сервером додатків і базою даних.

Після того як драйвер встановить з'єднання, виконується запит до бази, в результаті чого формується набір записів. Набір записів є безліч даних, отриманих з однієї або декількох таблиць бази даних. Набір записів повертається сервера додатків, який використовує отримані дані для формування сторінки.

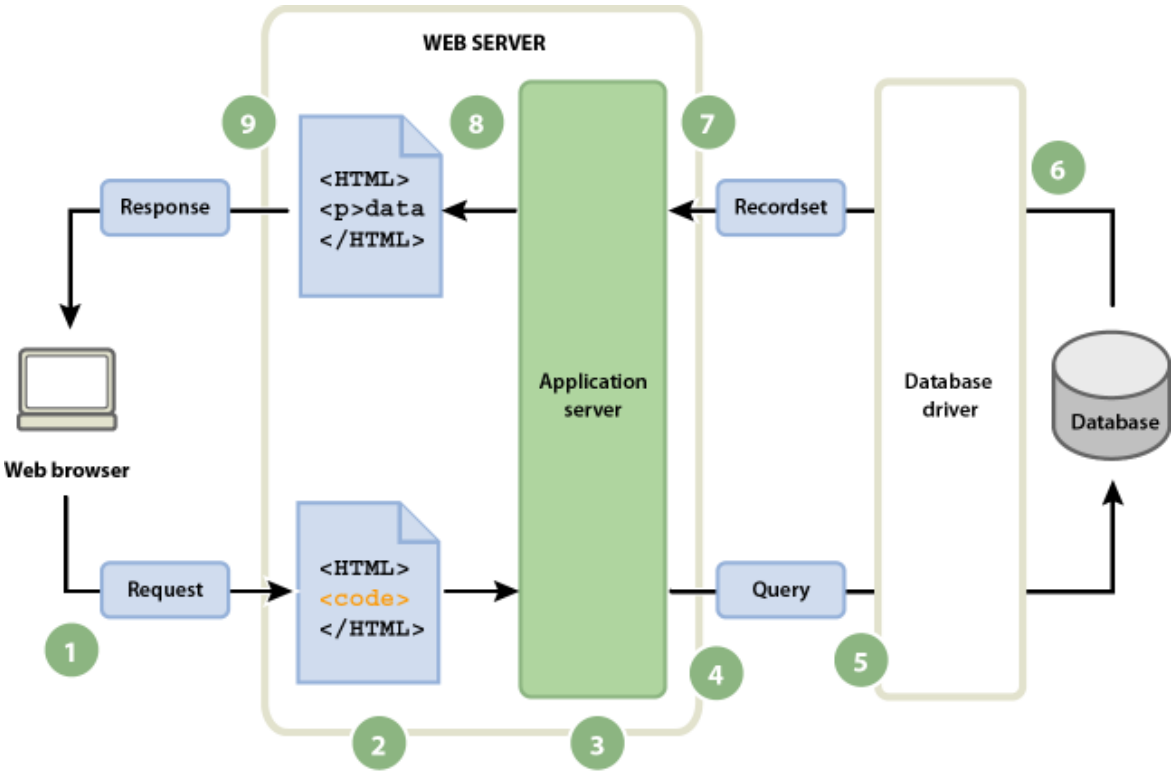

Рис. 2.2.3. Запит статичної сторінки

Також слід зауважити, що у веб-додатках можуть використовуватися будь-які серверні технології. Оскільки значна частина веб-додатку переміщається в браузер, вимоги до сервера можна істотно послабити.

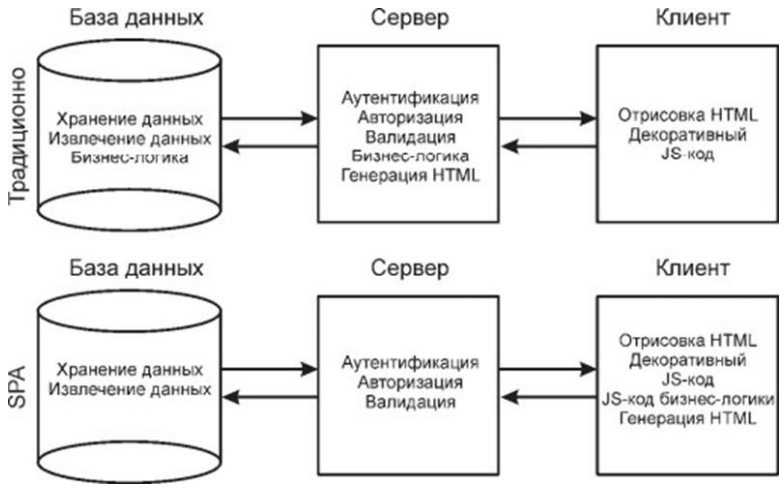

Рис. 2.2.4. Схема порівняння роботи веб-додатку і традиційного підходу до створення веб-сайтів

На рисунку 2.2.4. представлені відмінності між підходами при реалізації звичайного веб-сайту і веб-додатку. Як ми бачимо зі схеми, у веб-додатку основна частина роботи з даними переходить з сервера на клієнт.

Веб-додаток промальовується як персональне додаток. Воно перемальовує лише ті частини інтерфейсу, які змінилися, і лише тоді, коли це необхідно. Навпаки, традиційний сайт перемальовує всю сторінку у відповідь на різні дії користувача, що призводить до затримок і «миготіння», оскільки браузер повинен отримати сторінку від сервера і намалювати її на екрані.

Якщо сторінка велика, сервер зайнятий або з'єднання з Інтернетом повільне, то затримка може скласти кілька секунд. Це великий недолік, в порівнянні з швидкодією і миттєвим зворотним зв'язком веб-додтків.

Веб-додаток може реагувати як персональне додаток. Воно мінімізує час реакції за рахунок того, що переносить робочі (тимчасові) дані і частина обробки з сервера в браузер.

У розпорядженні веб-додатку є дані і бізнес-логіка, необхідні для прийняття більшості рішень локально, а значить, швидко. Лише автентифікація користувача, валідація та постійне зберігання даних повинні залишитися на сервері. У разі традиційного сайту велика частина логіки додатка знаходиться на сервері, тому, щоб отримати відповідь на свої дії, користувач повинен дочекатися завершення циклу запит-відповідь-оновлення. Це може зайняти кілька секунд, тоді як реакція веб-додатку майже миттєва.

Веб-додаток може повідомляти користувача про свій стан, як і персональне додаток. Якщо веб-додаток все-таки має чекати відповіді сервера, то воно може динамічно оновлювати індикатор ходу виконання або зайнятості, щоб користувач не лякався затримки.

Веб-додаток, як і сайт, майже завжди доступний. На відміну від більшості переносних програм, користувач може звернутися до веб-додатку, маючи лише з'єднання з Інтернетом і пристойний браузер. В даний час все це є на смартфонах, планшетах, телевізорах, ноутбуках і настільних ПК. Немає складнощів з підтриманням актуальності численних програм. Персональний додаток зазвичай необхідно завантажити, а потім встановити з правами адміністратора, а інтервал між випуском версій може становити кілька місяців або навіть років.

Веб-додаток, як і сайт, працює на різних платформах. На відміну від більшості переносних програм, добре написаний веб-додаток може працювати в будь-якій операційній системі, де є сучасний браузер з підтримкою HTML5. Зазвичай ця особливість вважається перевагою для розробника, але вона не менш важлива численним користувачам, що працюють з кількома пристроями, скажімо, з Windows на роботі, з Mac'ом вдома, з сервером під управлінням Linux і з телефоном Android.

Веб-додаток може запропонувати найкраще з обох світів - миттєву реакцію персонального докладання поряд з перенесенням і доступністю веб-сайту. JavaScript веб-додатків є більш ніж мільярду пристроїв, які підтримують сучасні браузери і не потребують сторонніх підключаються модулях. Приклавши трохи зусиль, можна зробити так, що воно буде працювати на настільних ПК, планшетах і смартфонах з різними операційними системами. Веб-додаток легко оновлювати і поширювати, зазвичай це не вимагає ніяких дій з боку користувача.

Саме перераховані плюси односторінкового додатки перед звичайним сайтом роблять його оптимальним вибором для реалізації даного проекту.

2.3. Вибір інструментів

Front-End

Angular 2+

Переваги:

Головна перевага Angular 2+ – це його популярність. Можна говорити про те, що з ним пов'язано ім'я компанії Google і це впливає на те, як його сприймають. Angular 1 швидко став популярним так як ті, хто прийшов з інших середовищ розробки виявили в ньому знайомий шаблон MVC для створення веб-додатків. Після модернізації Angular 1 і перепроєктування деяких частин фреймворка, Angular 2+ став одним із самих популярних фреймворків для розроблення веб-додатків. На ринку є серйозна потреба в Angular-розробниках. Крім того, це – один з небагатьох фреймворків, у якого є офіційний набір багатих можливостями компонентів для створення користувацьких інтерфейсів.

Недоліки:

Angular зосереджений на створенні призначених для користувача інтерфейсів односторінкових додатків і не відповідає потребам розробників більших проектів. Це може привести до складності підтримки проектів, якщо базові принципи, на яких вони засновані, що не були чітко сформульовані в самому початку їх розробки. На практиці розробникам доводиться витрачати багато зусиль, щоб змусити додаток на Angular робити те, що не є частиною фреймворка. Це, крім того, знижує інтерес розробників до TypeScript, на якому написаний фреймворк.

React + Redux

Переваги:

Основна перевага React і Redux полягає в їх простоті і в тому, що вони спрямовані на вирішення однієї задачі, на розробку інтерфейсів. Якщо треба щось, що робить щось одне, але робить це добре, то можна сказати, що обидві бібліотеки відмінно роблять те, чого від них чекають. У той час, як для кого-то підхід, пов'язаний з використанням контейнера стану може здатися незнайомим, більшість розробників можуть легко розібратися в цій концепції і зрозуміти переваги архітектури, заснованої на однонаправленому потоці даних та як такий підхід може спростити додатки зі складними призначеними для користувача інтерфейсами.

Недоліки:

Найбільші мінуси React і Redux полягають не в особливостях реалізації того, що вони вміють, а в тому, чого вони не можуть. Для того, щоб створити складне веб-додаток, вам знадобиться багато інших технологій. Як тільки ви відійдете від основних функцій React, Redux і пари інших бібліотек, ви зіткнетеся з незліченною кількістю рішень та шаблонів, які іноді легко інтегрувати в додаток, а іноді - ні.

Отже, так як React і Redux - бібліотеки, які зосереджені на вирішенні вузького кола спеціалізованих завдань, недосвідчені розробники можуть створити непідтримуваний продукт. Навіть досвідчені розробники можуть зіткнутися з тим, що недолік чіткого архітектурного планування рішення або строгих правил на початку розробки можуть дуже неприємно позначитися на проекті в майбутньому.

Vue.js

Переваги:

Ймовірно, головний плюс цього фреймворка полягає в можливості його поступового впровадження. Vue відрізняється зрозумілою і раціональної архітектурою, яку нескладно освоїти і просто застосовувати на практиці. Існує згуртоване співтовариство ентузіастів і сторонні проекти, які роблять Vue.js ще цікавіше. Крім того, різні розробки, орієнтовані на Vue, досить просто поєднувати в більш складних рішення при створенні нових проектів.

Недоліки:

Бажання якось викрутитися між ідеями додатків, заснованих на шаблоні MVC, і додатках, заснованих на контейнерах стану, може заплутати. Здається, що у розробників фреймворка є прагнення зробити все якісно, при цьому не даючи переваги одним шаблоном розробки додатків перед іншим. Нам здається, що це, як мінімум, збиває з пантелику тих, хто шукає в Vue.js платформу для повномасштабного веб-рішення, і може привести до застосування різних шаблонів, що, в підсумку, ускладнить підтримку програми.

Одна з головних проблем Vue.js полягає в тому, що проект залежить від однієї людини. Зрозуміло, що інші фреймворки теж від кого-то залежать, але зазвичай це - організації. Навколо Vue.js склалося велике співтовариство, тут є з безліччю інноваційних додаткових проектів, але розробка ядра цілком лежить на плечах єдиного розробника.

Ember

Переваги:

Мабуть, Ember.js - це найбільш строго організований фреймворк з тих, що набули поширення. У цьому - його головний плюс. При розробці проектів на Ember існують цілком певні правильні способи зробити щось, які зазвичай є єдино можливими.

Ember більше схожий на платформу, на якийсь продукт, від постачальника якого можна було б очікувати довготривалу підтримку і обслуговування. Ember.js дає просунуту систему управління версіями своєї платформи, інструменти для переходу на нові версії, і чіткі керівництва та засоби по обходу застарілих API. Мабуть, Ember має повне право називатися зрілим фреймворком.

За роки існування Ember.js показав, що його команда може підтримувати фреймворк і своєчасно інтегрувати в нього сучасні стандарти, але, в той же час, не закидати старі браузери.

Ember має чітку і раціональну архітектуру, яка підходить для розробки складних веб-додатків.

Недоліки:

Головний мінус Ember полягає в тому ж самому, в чому криється його головний плюс. Мова йде про жорстку структуру проектів, створених з його використанням.

Хоча його спільнота відкрито і доброзичливо ставиться до пропозицій щодо вдосконалення Ember, при розробці проектів на цьому фреймворку завжди існує правильна послідовність дій, запропонована

самої архітектурою фреймворка. Відхід в сторону може вилитися в проблеми.

Певні складнощі може викликати і те, що тут немає стандартного набору елементів призначеного для користувача інтерфейсу, тому доводиться користуватися такими наборами сторонніх розробників.

Ймовірно, може здатися незручним і те, що ці набори не можна назвати всеосяжними, відсутні компоненти доведеться шукати або створювати самостійно. Так як Ember.js не регламентує взаємодію з DOM, ви можете опинитися в ситуації, коли у вашому проекті є неоднорідні компоненти, з яких побудований призначений для користувача інтерфейс, яким не дуже зручно управляти.

Після ознайомлення з недоліками та перевагами усіх найпопулярніших фреймворків, мною був обраний Angular.

Angular – це клієнтський MVC-фреймворк, написаний на JavaScript. Він виконується в веб-браузері і надає величезну допомогу в створенні сучасних, односторінкових веб-додатків, що використовують технологію AJAX. Це багатоцільовий фреймворк, але особливо яскраво його особливості проявляються при реалізації веб-додатків типу CRUD.

Angular зовсім недавно поповнив сімейство клієнтських MVC-фреймворків, проте йому вдалося привернути до себе увагу, в основному завдяки своїй інноваційній системі шаблонів, простоті розробки з його використанням і застосування надійних інженерних рішень. Його система шаблонів дійсно багато в чому унікальна:

• В якості мови шаблонів в ній використовується мова розмітки HTML.

• Вона не вимагає явно оновлювати дерево DOM, так як Angular здатний стежити за діями користувача, подіями браузера і змінами в моделі, і вчасно виявляти, коли і який шаблон потрібно оновити.

• Має вельми цікаву і розширювану підсистему компонентів, і має можливість «навчати» браузер розпізнаванню і правильної інтерпретації нових тегів і атрибутів HTML.

Підсистема шаблонів є, мабуть, найпомітнішою частиною Angular, але було б неправильним вважати, що Angular – це звичайний фреймворк, що включає в себе кілька утиліт і служб, зазвичай необхідних для односторінкових веб-додатків.

Angular має кілька прихованих можливостей, механізм впровадження залежностей (Dependency Injection, DI) і сильний акцент на тестування. Вбудована підтримка DI істотно спрощує збірку веб-додатків з невеликих, надійно протестованих служб. Архітектура фреймворка і оточуючих його інструментів сприяє застосуванню тестування на всіх етапах розробки.

Angular - це проект з відкритим вихідним кодом, який можна знайти на GitHub (https://github.com/angular/angular.js), і ліцензується компанією Google, Inc. на умовах ліцензії MIT.

У стандартній формі HTML значенням елемента вводу є значення, яке буде відправлено на сервер в складі форми (Рис. 2.2.5.).

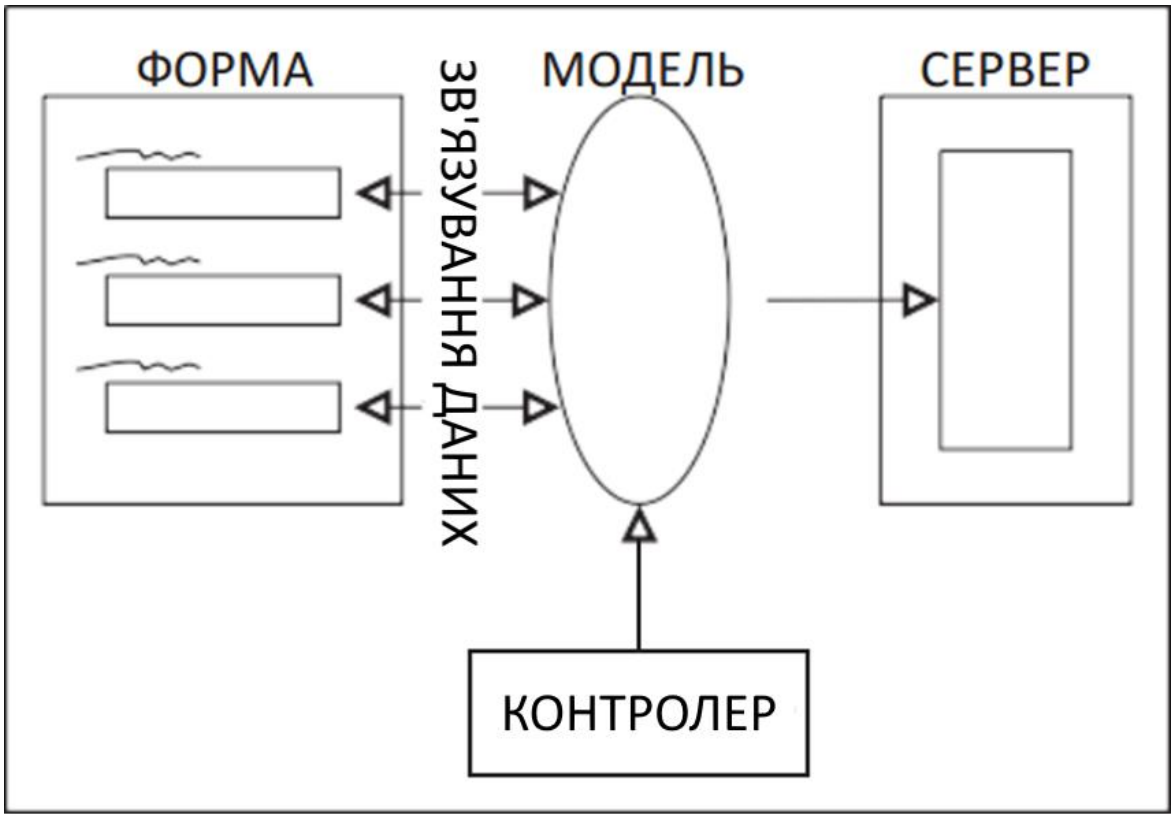

Рис. 2.2.5. Відокремлення моделі від представлення

Основна концепція фреймворку Angular складається в архітектурному шаблоні MVC (Model-View-Controller). Модель-Представлення-Контролер (або MVVM, що позначає модель-Уявлення-Представлення-Контролер, який дуже схожий) розвивається як спосіб поділу логічних блоків при розробці великих додатків. Це дає розробникам відправну точку в ухваленні рішення про те, як і де можна розділити обов’язки. Архітектурний шаблон MVC ділить додаток на три окремі модульні частини:

- Модель є рушійною силою програми. Це, як правило, дані програми, зазвичай отримані з сервера. Будь-призначений для користувача інтерфейс з даними, які бачить користувач, отриманий з моделі або підмножини моделі.

- Представлення – це призначений для користувача інтерфейс, який користувач бачить і взаємодіє. Він динамічний і генерується на основі поточної моделі програми.
- Контролер – це рівень бізнес-логіки та інтерфейсу, який виконує такі дії, як вибірка даних, і приймає рішення, наприклад, як представити модель, які частини її відображати і т. д.

Angular буде корисний у тих випадках, коли потрібне створення великого інтерфейсу користувача або повноцінного односторінкового додатку зі складною архітектурою і розгалуженими зв’язками. Таким проектом якраз і є клієнтська частина нашого застосування.

Back-End

Так як в нашому продукті є можливість зберігати дані – він передбачає наявність серверної частини. Проте функціонал нашого продукту зведений до мінімуму, тому сутностей в базі даних буде небагато.

Користувачі, як такі в структурі відсутні. Опитувані мають лише ПІБ та ключ за яким входять до опитування. Що стосується адміністратора – то він представлений у вигляді єдиного екземпляру, так як система не передбачає розділення на ролі користувачів адмін-панелі.

Залишаються сутності опитувань та питань, в яких буде невизначена кількість здебільшого однотипних атрибутів та кілька полів для параметрів опитування/питання.

Виходячи з цього буде логічним зробити висновок, що замість розроблення серверної частини та написання API, доцільніше скористуватись готовими сервісами, які допоможуть зберігати дані та передавати їх на клієнт.

Firebase

Серед таких сервісів мій вибір одразу пав на Firebase. Він є безумовно приголомшливим у виконанні, реалізації та експлуатації. Для нашого продукту Firebase ідеально підходить і виконує усі функції серверної частини. При цьому швидкодія і оптимальність майже не поступаються повноцінній серверній частині.

Firebase служить базою даних, яка змінюється в реальному часі і зберігає дані в JSON. Будь-які зміни в базі даних відразу синхронізуються між усіма клієнтами, або девайсами, які використовують одну і ту ж базу даних. Іншими словами, оновлення в Firebase відбуваються миттєво.

Разом зі сховищем, Firebase також надає призначену для користувача аутентифікацію, і тому всі дані передаються через захищене з'єднання SSL. Ми можемо вибрати будь-яку комбінацію email і пароля для аутентифікації, будь то Facebook, Twitter, GitHub, Google, або щось інше.

У Firebase є SDK для iOS, Android і JavaScript. Всі платформи можуть використовувати одну базу даних, що стане необхідним у разі розширення.

Також слід зазначити, що цей сервіс може слугувати хостингом для нашого веб-додатку. Таким чином ми економимо максимум ресурсів.

А що саме головне цей сервіс є абсолютно безкоштовним. В безкоштовному пакеті Firebase може обробляти до 100 одночасних з'єднань, чого від звичайних безкоштовних хостингів очікувати не доводиться. Цього цілком достатньо для нашого продукту й навіть для роботи популярного додатку.

Переваги:

- Безкоштовний сервіс;

- Скорочує час розроблення за рахунок непотрібності писати повноцінну серверну частину, проєктувати БД, та розробляти API;
- Є хостингом для клієнт-частини;
- Має багато користувачів, а значить і багату кількість рішень для будь-яких потреб

Недоліки:

- Не реляційна БД не дозволить зберігати швидкодію роботи сервісу при великих об’ємах даних;
- Дубляж даних та не структурована інформація;

JSON

JSON - текстовий формат даних, схожий за синтаксисом на об'єкт JavaScript. Незважаючи на це, його можна використовувати незалежно від JavaScript, і багато середовищ програмування мають можливість читати (аналізувати) і генерувати JSON.

JSON існує як рядок – це корисно, коли ви хочете передавати дані по мережі. Він повинен бути перетворений в власний об'єкт JavaScript, якщо ви хочете отримати доступ до даних. Це не велика проблема. JavaScript надає глобальний об'єкт JSON, який має методи для перетворення між ними.

Об'єкт JSON може бути збережений у власному файлі, який в основному є текстовий файл з розширенням .json і MIME type application / json.

В JSON можна включати одні й ті ж базові типи даних всередині, як ви можете і в стандартному об'єкті JavaScript - рядки, числа, масиви,

булеві і інші об'єктні літерали. Це дозволяє побудувати ієрархію даних, наприклад, так:

```
{
  "orderID": 12345,
  "shopperName": "Ivan Ivanov",
  "shopperEmail": "ivanov@example.com",
  "contents": [
    {
      "productID": 34,
      "productName": "Super product",
      "quantity": 1
    },
    {
      "productID": 56,
      "productName": "Wonderful product",
      "quantity": 3
    }
  ],
  "orderCompleted": true
}
```

Рис. 2.2.6. JSON

Особливості:

- JSON - це чисто формат даних - він містить тільки властивості, без методів.
- JSON вимагає подвійних лапок, які будуть використовуватися навколо рядків і імен властивостей. Одиничні лапки недійсні.
- Навіть одна недоречна кома або двокрапка можуть привести до збою JSON-файлу і не працювати. Ви повинні бути обережні, щоб перевірити будь-які дані, які ви намагаєтеся використовувати (хоча згенерований комп'ютером JSON з меншою ймовірністю включає помилки, якщо програма генератора працює правильно). Ви можете перевірити JSON за допомогою програми на кшталт JSONLint.
- JSON може приймати форму будь-якого типу даних, допустимого для включення в JSON, а не тільки масивів або

об'єктів. Так, наприклад, один рядок або номер будуть дійсним об'єктом JSON.

- На відміну від коду JavaScript, в якому властивості об'єкта можуть не обертатися в подвійні лапки, в JSON в якості властивостей можуть використовуватися тільки рядки загорнуті в подвійні лапки.

Після того як було проаналізовано аналоги, визначено вимоги до додатку, проведено аналіз сучасних технологій розробки веб-додатків та обрано найбільш доцільніші для реалізації продукту інструменти, можна розпочати роботу над власне розробленням веб-додатку.

РОЗДІЛ 3.
РОЗРОБЛЕННЯ ВЕБ-ДОДАТКУ

3.1. Структура додатку

Веб-додаток поділяється на дві частини, які відкриваються користувачу в залежності від введених їм даних на сторінці авторизації. Після авторизації користувачу відкривається відповідний екран та надається веб-токен, з яким він в подальшому звертається до сервера за необхідними даними.

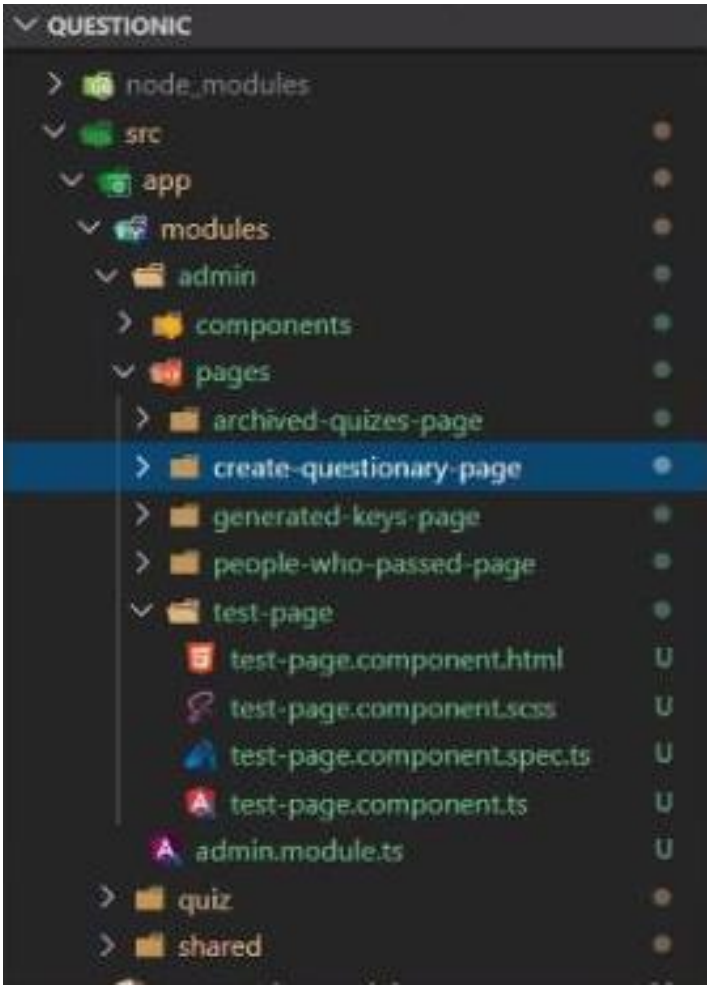

Рис. 3.1. Структура веб-додатку

Перша частина – це адмін-панель (папка «admin»), а друга частина – це опитування (папка «quiz»). До кожної частини належать свої екрани. Опитування мають лише один екран.

Адмін-панель складається з наступних екранів:

- списку опитувань (папка «test-page»);
- перегляду опитування (папка «archived-quizes-page»);
- створення опитування (папка «create-questionary-page»);
- генерації ключів (папка «generated-keys-page»);
- списку опитаних (папка «people-who-passed-page»).

Кожний екран складається з наступних файлів:

- *.component.html;
- *.component.scss;
- *.component.spec.ts;
- *.component.ts;

Файл *.component.html в кожній папці є файлом розмітки до основного HTML файлу проекту. Основний HTML файл проекту виступає у вигляді контейнеру, який доповнюють HTML файлом відкритого екрану. Таким чином сторінка не перезавантажується а просто перемальовується з переходом від одного екрана до іншого. В *.scss файлах ми маємо високо абстрактні інструкції стилів до кожної сторінки.

Саме завдяки правильному опису цих двох файлів ми досягаємо адаптивності нашого веб-додатку.

У *.ts файлах ми зберігаємо всі наші змінні з якими працюємо в клієнт частині, перед тим як відправити данні на сервер, або навпаки зберігаємо в цих змінних данні отримані з серверу. Також в них описані функції, що відповідають за динамічність нашого сайту.

На такій структурі і побудована робота веб-додатку та вміст і функціонал усіх екранів в ньому.

3.2. Структура серверу

Так як ми використовуємо Firebase, всі наші данні зберігаються в JSON файлах. Таким чином головні сутності в нашій БД виглядають наступним чином:

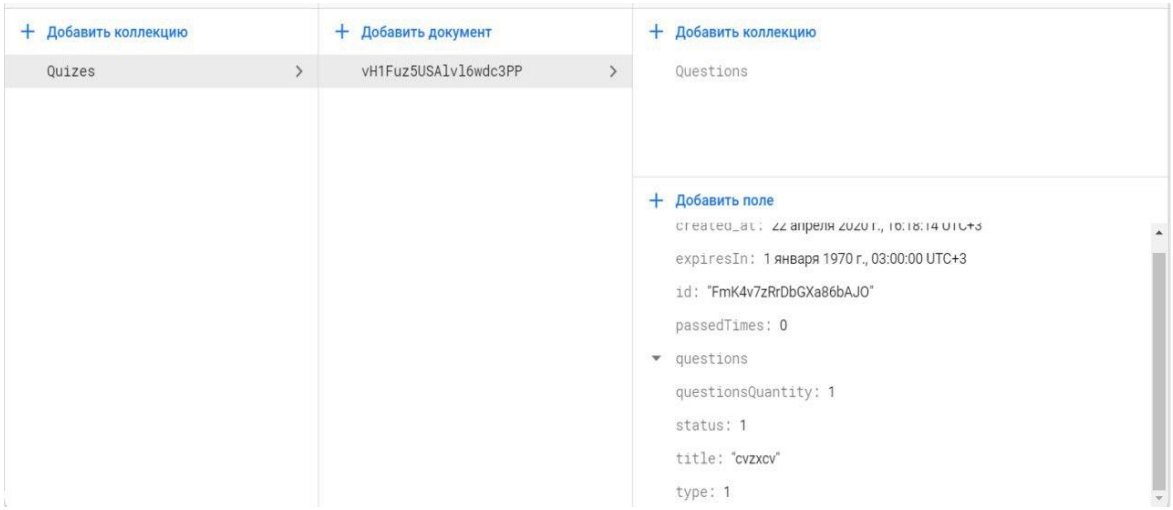

Рис. 3.2.1 Опитування (сервер)

Це екземпляр опитування (Рис. 3.2.) на нашому сервері, зберігається в JSON файлі. Ця «сутність» містить наступні атрибути:

- Первинний ключ (id) – генерується випадковим чином;
- Дата створення (create_at) - береться дата й час публікації опитування;
- Дата завершення опитування (expiresIn) – встановлюється користувачем;
- Статус (status) – 0 чернетка, 1 активне, 2 архів;
- Назва (title) – встановлюється користувачем;
- Тип (type) – 0 тестування, 1 анкетування;
- Вкладена колекція (questions) – масив з питаннями.

У вкладеній колекції знаходяться екземпляри питань, у вигляді масиву, кожний елемент якої має структуру аналогічну структурі опитування.

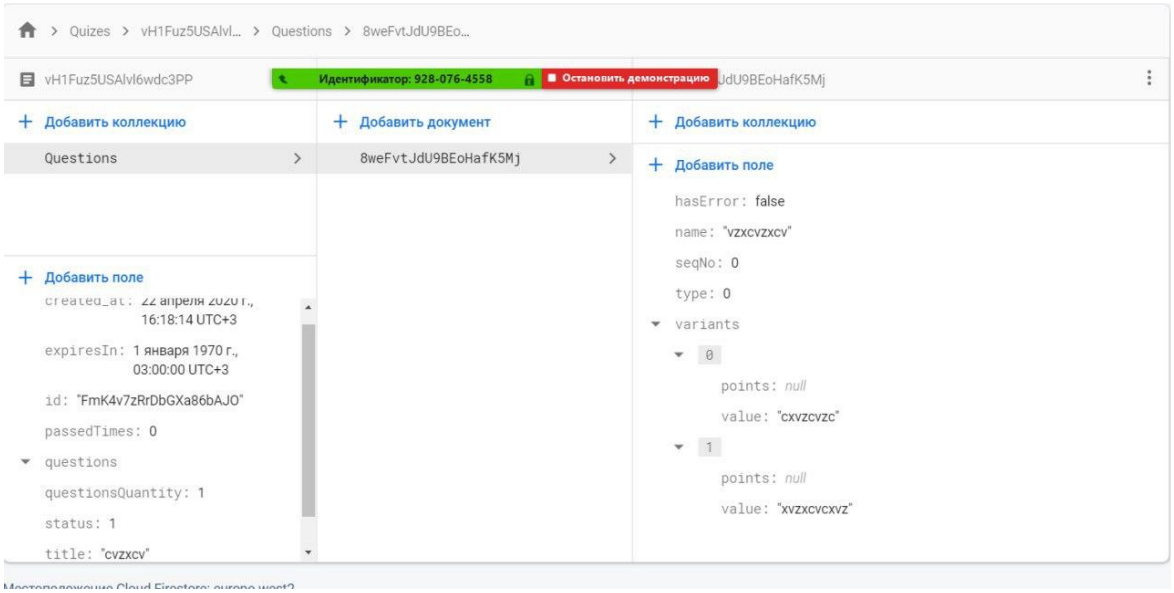

Рис. 3.2.2 Питання (сервер)

Опитування потрапляє в базу даних при збереженні або публікації. При чому при публікації створюється дублікати опитуванні в кількості, яка дорівнює кількості введених адміністратором ключів. На жаль це єдиний спосіб закріпити до опитування згенеровані ключі, по яким потім буде проходити опитування користувач.

Авторизація адміністратора відбувається порівнянням введених в поля значень зі статичними, що введені на сервері, так як в системі передбачається одна роль створювача опитувань.

Авторизація опитуваних відбувається порівнянням введеного ключа з усіма існуючими, після чого (якщо введений ключ присутній в базі і його статус – «невикористаний»), йде перевірка на тип опитування, перевірка ПІБ (якщо тип опитування – тестування) та передача файл JSON на клієнт, із вмістом опитування. Після завершення опитування цей файл відправляється відредагованим (з вибраними відповідями) назад на сервер, де перезаписується в БД.

ВИСНОВКИ

У ході дипломної роботи була поставлена мета спроектувати та розробити веб-додаток для здійснення педагогічного тестування.

Досягнення зазначеної мети здійснюється шляхом вирішення таких **основних завдань**:

- Аналіз аналогів.
- Визначення вимог до веб-додатку.
- Складання діаграм прецендентів .
- Розглядання сучасних підходів та технологій розробки мобільних додатків.
- Розроблення веб-додатку.

Розроблений веб-додаток відповідає всім визначеним нами вимогам та готовий до експлуатації першою версією нашого сервісу. Додаток має перспективи подальшого розвитку. Пройти опитування за наданим вам ключем або створити опитування чи переглянути його можна за цим посиланням:

https://questionic-860b3.firebaseapp.com/

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Как работают веб-приложения [Електронний ресурс] - Режим доступу: https://habr.com/ru/post/450282/ССЫЛКА
2) РАЗРАБОТКА WEB-ПРИЛОЖЕНИЙ [Електронний ресурс] - Режим доступу: https://infoshell.ru/blog/razrabotka-veb-prilozhenij/
3) Веб-приложение [Електронний ресурс] - Режим доступу: https://semantica.in/blog/veb-prilozhenie.html
4) Website и Web Application: в чем разница? [Електронний ресурс] - Режим доступу: https://dinarys.com/ru/blog/website-vs.-webapplication
5) Разработка мобильных и веб-приложений: что является лучшим решением [Електронний ресурс] - Режим доступу: https://smartum.pro/ru/blog-ru/razrabotka-mobilnykh-i-web-prilozheniy/
6) Формат JSON, метод toJSON [Електронний ресурс] - Режим доступу: https://learn.javascript.ru/json
7) Работа с JSON [Електронний ресурс] - Режим доступу: https://developer.mozilla.org/ru/docs/Learn/JavaScript/Объекты/JSON
8) Введение в JSON [Електронний ресурс] - Режим доступу: https://www.json.org/json-ru.html
9) Что такое NoSQL? [Електронний ресурс] - Режим доступу: https://aws.amazon.com/ru/nosql/
10) SQL или NoSQL — вот в чём вопрос [Електронний ресурс] - Режим доступу: https://habr.com/ru/company/ruvds/blog/324936/

11) Нереляционные данные и базы данных NoSQL [Електронний ресурс] - Режим доступу: https://docs.microsoft.com/ru-ru/azure/architecture/data-guide/big-data/non-relational-data
12) Нереляционные базы данных [Електронний ресурс] - Режим доступу: https://spravochnick.ru/bazy_dannyh/nerelyacionnye_bazy_dannyh/
13) SQL и NoSQL: разбираемся в основных моделях баз данных [Електронний ресурс] - Режим доступу: https://tproger.ru/translations/sql-nosql-database-models/
14) СРАВНЕНИЕ РЕЛЯЦИОННЫХ И НЕ РЕЛЯЦИОННЫХ (NOSQL) БАЗ ДАННЫХ [Електронний ресурс] - Режим доступу: https://sibac.info/studconf/science/xliv/106548
15) React [Електронний ресурс] - Режим доступу: https://ru.react.js.org
16) Основы React.js [Електронний ресурс] - Режим доступу: https://learn.javascript.ru/screencast/react
17) Понимаем React за 5 минут [Електронний ресурс] - Режим доступу: https://medium.com/@stasonmars/понимаем-react-за-5-минут-2611122c6fbf
18) Введение в Angular [Електронний ресурс] - Режим доступу: https://metanit.com/web/angular2/1.1.php
19) Удивительный Angular [Електронний ресурс] - Режим доступу: https://habr.com/ru/post/348818/
20) Скринкаст по Angular [Електронний ресурс] - Режим доступу: https://learn.javascript.ru/screencast/angular
21) AngularJS [Електронний ресурс] - Режим доступу: https://angularjs.org

22) Angular застосунок своїми руками за 2 години [Електронний ресурс] - Режим доступу: https://codeguida.com/post/1509

23) What is Firebase? [Електронний ресурс] - Режим доступу: https://howtofirebase.com/what-is-firebase-fcb8614ba442

24) Введение в Firebase: пишем простое социальное приложение на Swift [Електронний ресурс] - Режим доступу: https://habr.com/ru/post/277941/

25) What is Firebase? The complete story, abridged. [Електронний ресурс] - Режим доступу: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

26) React Native Firebase [Електронний ресурс] - Режим доступу: https://rnfirebase.io

27) Introduction to Firebase [Електронний ресурс] - Режим доступу: https://hackernoon.com/introduction-to-firebase-218a23186cd7

28) Firebase [Електронний ресурс] - Режим доступу: https://stackshare.io/firebase

29) Using Firebase [Електронний ресурс] - Режим доступу: https://docs.expo.io/guides/using-firebase/?redirected

30) Firebase: прощание с иллюзиями [Електронний ресурс] - Режим доступу: https://habr.com/ru/company/livetyping/blog/320040/

31) Этапы проектирования данных [Електронний ресурс] - Режим доступу: http://www.mstu.edu.ru/study/materials/zelenkov/ch_5_1.html

ДОДАТКИ

Додаток 1

Додаток 1

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Шувалов Дмитрій Кирилович,
учасник(ця) освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:
- вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
- принципів та правил академічної доброчесності;
- нульової толерантності до академічного плагіату;
- моральних норм та правил етичної поведінки;
- толерантного ставлення до інших;
- дотримуватися високого рівня культури спілкування;

– надавати згоду на:
- безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
- оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
- використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науково-дослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;

– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

24.09.2020 (дата) _____ (підпис) Дмитрій Шувалов (ім'я, прізвище)