**Техническое Задание (ТЗ)**

**1. Введение**

1.1 Название проекта

Вопросы внутрь. Самоанализ.

1.2. Описание проекта

Приложение для смартфона под операционную система Android, предлагающее пользователю задать себе вопросы в течение дня, создавать и сохранять историю ответов, давать ответ цветом и получать цитаты с сайта Rapid через Quotes API.

1.3. Цели проекта

Создать простое, минималистичное приложение, помогающее людям углубится в самоанализ и рефлексию для улучшения качества жизни.

1.4. Сроки работы:

|  |  |  |
| --- | --- | --- |
| Этапы работы | Начало | Окончание |
| Планирование и анализ | 01.10.2024 | 07.10.2024 |
| Дизайн и проектирование | 08.10.2024 | 21.10.2024 |
| Разработка | 22.10.2024 | 03.12.2024 |
| Тестирование и валидация | 04.12.2024 | 17.12.2024 |
| Развертывание и публикация | 18.12.2024 | 24.12.2024 |
| Поддержка и обслуживание | 25.05.2024 |  |

*Табл. 1. Сроки выполнения работы*

**2. Функциональные Требования**

2.1. Основной функционал

* автоматическая генерация вопросов из базы данных
* Возможность сохранения ответа на вопрос, которые отображаются в истори вопросов
* История ответов на вопросы в текстовом формате, и формате «ответ цветом»
* При каждом входе в раздел «ответ вопросом» поле ввода остается пусты
* Возможность ответа с помощью цвета из палитры
* Сохранение ответов цветом к вопросу с записью в базу данных
* Ответ цветом отображается в истории
* Функция «подсказка дня» выводит цитаты с сайта Rapid на русском языке через внешнее Quotes API
* При повторном входе в функцию «подсказка дня» цитаты обновляются

2.2. Дополнительные функции

* настраивание частоты изменения вопроса: 1, 3, 12 и 24 часа.
* Изменение основного фонового цвета интерфейса из палитры цветов
* наличие виджетов, которые выводятся на главный экран в заблокированном и разблокированном состоянии устройств
* При нажатие пользователем на виджет, он попадает на главную страницу приложения

**3. Нефункциональные Требования**

3.1. Требования к производительности

* время отклика: не более 2 секунд
* Время загрузки: не более 3 секунд
* Время обработки запроса к базе данных не более 1 секунды
* Время обработки запросов через API не более 2 секунд
* Приложение должно корректно работать в фоновом режиме без значительных потреблений ресурсов устройства

3.2 Требования к безопасности

* Сохранение данных: ответы пользователей хранятся безопасно в базе данных на сервере
* Идентификатор устройства: используется для привязки данных к конкретному устройству
* Конфиденциальность: данные передаются по защищенному каналу HTTPS

3.3. Надежность

* Доступность приложения должна быть 99.9% времени
* Данные должны сохраняться без потерь, даже в случае сбоев, перезапуска или отключения.
* База данных должна справляться со всеми входными запросами и корректно добавлять данные

3.4. Удобство использования

* Интерфейс приложения должен быть интуитивно понятен пользователю
* Навигация по приложению должна быть простой, логичной и структурированной
* К приложению должна прилагаться документация об эксплуатации, которая поможет пользователю быстро освоить все функции

3.5. Совместимость

* приложение должно корректно работать и быть совместимы с версиями Android 6.0 и выше
* Приложение должно корректно работать с разным разрешением экранов
* Приложение имеет только один тип ориентации - портрет

3.6. Масштабируемость

* Архитектура приложения должна быть построена так, чтобы в приложения могли быть добавлены новые функции
* Приложение должно иметь возможность обслуживать увеличивающееся количество пользователей

3.7. Мобильность

* Базовый режим приложения должен иметь функцию записи и сохранения ответов с их историей
* Базовый режим приложения должен поддерживать функцию виджетов
* Должна быть доступна синхронизация данных при подключению к интернету сразу, без ожидания инициации от AlarmManager

**4. Стек Технологий**

4.1. Фронтенд:

* Языки программирования - Kotlin
* Инструменты разработки - Android Studio

4.2. Бэкенд:

* Платформа: Node.js
* Фреймворк: Express

4.3. База данных

* База данных: PostgreSQL
* Инструмент управления: pgAdmin

4.4. API

* Тип: RESTful API

4.5. Дизайн

* Инструмент: Figma

4.6. Контроль версий

* Инструменты: Git и GitHub

**5. Архитектура Системы**

5.1 компоненты системы

  5.1.2. Компоненты

    5.1.2.1.Клиентское приложение (фронтенд):

* Все данные между клиентом и сервером передаются в формате JSON через HTTP-запросы

     Компоненты:

* AlarmManager - настройка автоматических запросов к серверу
* SharedPreferences - сохранение пользовательских настроек
* AppWidgetProvider - управление виджетами на главном экране

    Функции:

* Вопрос дня: запрашивает данные у сервера
* Ответ текстом: отправляет данные ответа на сервер
* Ответ цветом: отправляет данные ответа на сервер
* Подсказка дня: запрашивает данные у сервера
* Настройка частоты ответа: сохраняет выбранную частоту и настраивает AlarmManager
* Изменение цвета фона: сохраняет цвет и обновляет интерфейс
* Виджеты: отправляет запрос на обновление виджетов через AppWidgetProvider

   5.1.2.2. Сервер (бэкенд):

* Все данные между клиентом и сервером передаются в формате JSON через HTTP-запросы
* Все данные между сервером и базой данных происходят в формате SQL
* Все данные между сервером и происходят через API ( Quotes API) через HTTP запросы с форматом передачи JSON

    Функции:

* Обработка запросов: прием и обработка запросов от клиентского приложения
* Запросы к базе данных: извлекает данные и сохраняет новые записи в базе данных
* Запросы к внешнему API: отправляет запрос, получает данные от внешнего API и отправляет клиентскому приложению

  5.1.2.3. База данных

* Все данные между сервером и базой данных происходят в формате SQL

    Функции:

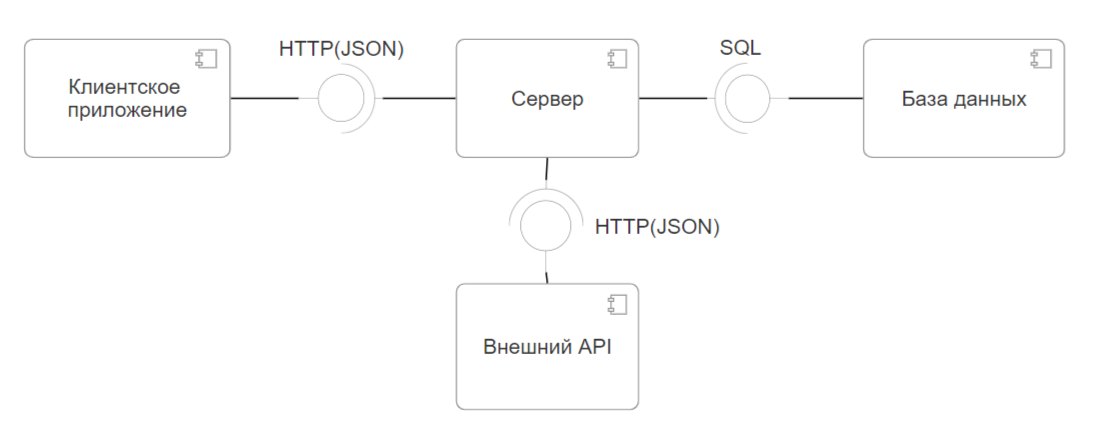
* Хранение данных: хранит пользовательские данные и данные приложения
* Обработка запросов:  выполняет SQL-запросы на чтение и запись данных

  5.1.2.4. Внешние API: Quotes API

* Все данные передаются через HTTP-запросы в формате JSON

    Функции:

* Получение цитаты: предоставить случайную цитату на русском языке

**

*Рис. 1. Диаграмма компонентов*

5.2. Взаимодействие компонентов

 5.2.1. Функция «вопрос дня»

  5.2.1.1. Клиентское приложение

* пользователь задает частоту вопросов в настройках приложения
* Приложение использует AlarmManager для создание автоматического запроса на сервер
* В заданное пользователем время AlarmManager инициирует запуск BroadcastReceiver
* BroadcastReceiver формирует запрос на получение от сервера данных о «вопросе дня” и отправляет запрос HTTP

  5.2.1.2. Сервер приложения

* Сервер принимает запрос от клиентского приложения
* Обрабатывает и формирует запрос к базе данных в формате SQL

  5.2.1.3.  база данных

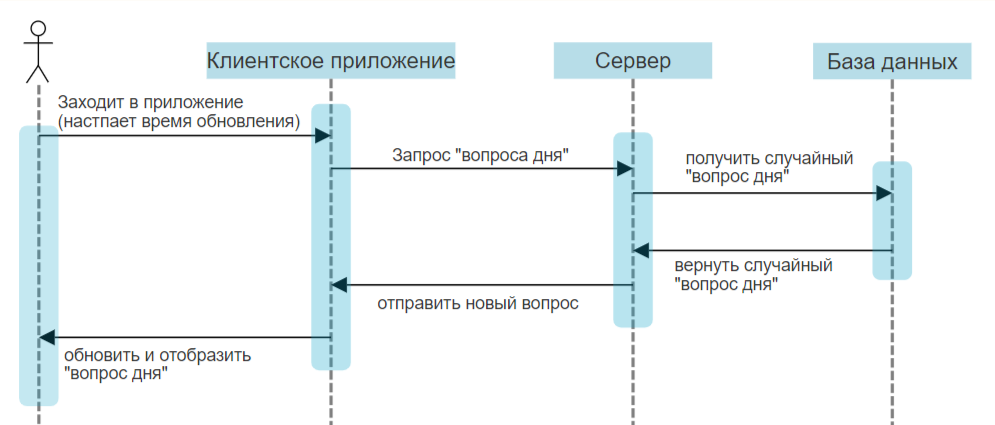
* база данных принимает запрос на получение вопроса
* Возвращается данные серверу

  5.2.1.4. Сервер приложения

* получает ответ от базы данных
* Обрабатывает и отправляет клиентскому приложению

  5.2.1.5. Клиентское приложение

* получает данные от сервера и обрабатывает их
* Обновляет отображение вопроса в приложение на главной странице приложения
* Обновляет отображение вопроса в виджета



*Рис. 2. Диаграмм последовательностей функции “порос дня”*

 5.2.2. Функция «ответ текстом»

  5.2.2.1. Клиентское приложение

* Пользователь нажимает кнопку на главном экране «ответ текстом
* Приложение отображает страницу с полем для ввода ответ
* Пользователь вводит текст
* Нажимает кнопку «сохранить» или «назад», что сохраняет ответ
* Клиентское приложение формирует HTTP-запрос с данными об ответе и идентификатором устройства

  5.2.2.2. Сервер приложения

* Сервер принимает HTTP-запрос
* Сервер извлекает из HTTP-запроса тело ответа
* Сервер формирует SQL-запрос для внесения в базу данных
* Отправляет запрос в базу данных

  5.2.2.3. База данных

* база данных принимает SQL запрос
* Вносит данные в базу
* Отправляет подтверждение о внесение данных на сервер

  5.2.2.4. Сервер

* Сервер получает данные об успешном сохранение ответа
* Сервер отправляет подтверждение о сохранение в формате JSON

  5.2.2.5. Клиентское приложение

* Получает JSON ответ от сервера и обрабатывает его
* Ответ остается видимым пользователю до выхода из раздела «ответ дня»
* После выхода из раздела «ответ дня» пользователем, поля ввода автоматически обнуляется и при повторном входе остается пустым.

5.2.3. Функция «ответ цветом»

  5.2.3.1. Клиентское приложение

* пользователь нажимает кнопку «ответ цветом»
* Приложение открывает вкладку с палитрой цветов и вопросом.
* Пользователь выбирает цвет из палитры и нажимает кнопку сохранить
* Приложение формирует HTTP-запрос с данными о выбранном цвете и идентификатором устройства на сервер в формате JSON

  5.2.3.2. Сервер

* получает данные от клиента
* Сервер извлекает ответ пользователя из запроса
* Формирует SQL запрос о внесение данных в базу и отправляет

  5.2.3.4. База данных

* получает запрос от сервера
* Вносит данные в базу
* Формирует ответ об успешном внесение данных

  5.2.3.5. Сервер

* получает ответ от базы данных
* Формирует подтверждение в формате JSON
* Отправляет клиенту

  5.2.3.6. Клиентское приложение

* получает от сервера ответ
* Отображает новый фон в виде цвета, выбранного пользователем
* Поле выбора цвета обнуляется после нажатия кнопки выход

5.2.4. Функция «история ответов»

  5.2.4.1. Клиентское приложение

* Пользователь нажимает кнопку «Истрия ответов»
* Приложение формирует HTTPS-запрос к серверу в формате JSON об извление данных для определенного идентификатора устройства

  5.2.4.2. Сервер

* Получает запрос от клиента
* Формирует два запроса на извлечение данных из таблиц «ответ цветом» и «ответ текстом» для идентификатора устройства
* Включает условия фильтрации по дате для базы данных
* Отправляет запрос в базу данных в формате SQL

  5.2.5.3. База данных

* База данных получает запрос
* Обрабатывает данные и возвращает ответ

  5.2.5.4. Сервер

* принимает ответ от базы данных
* Обрабатывает и формирует ответ в формате JSON
* Отправляет ответ клиенту

  5.2.5.5. Клиентское приложение

* получает данные от сервера об истории ответов
* Обрабатывает данные
* Выводит на страницу приложения список из его истории от последней сделанной записи да первой

5.2.6. Функция «подсказка дня»

  5.2.6.1. Клиентское приложение

* Клиент нажимает кнопку «подсказка дня»
* Приложение отправляет HTML-запрос в формате JSON на сервер

  5.2.6.2. Сервер

* получает запрос от клиента
* Формирует запрос к внешнему API Quotes API для получения случайной цитаты на русском языке в формате JSON

  5.2.6.3. Внешнее API Quotes API

* получает запрос от сервера
* Извлекает случайную цитату на русском языке из своей базы данных
* Отправляет в формате JSON ответ серверу

  5.2.6.4. Сервер

* получает ответ от внешнего API
* Обрабатывает и формирует ответ в формате JSON
* Отправляет клиенту ответ

  5.2.6.5. Клиентское приложение

* получает ответ от сервера
* Обрабатывает данные
* Отображает данные в виде текста на экране приложения

5.2.7. Функция «настройка частоты ответов»

  Клиентское приложение

* пользователь выбирает настройку «частота изменения ответов» во вкладке «настройки»
* Приложение отображает варианты выбора: 1, 3, 12, 24 часа
* Пользователь нажимает на выбранное время
* Пользователь нажимает стрелку «назад»
* В приложение AlarmManager формирует таймер
* AlarmManager автоматически инициирует запуск BroadcastReceiver

5.2.8. Функция «изменение цвета фона»

Клиентское приложение:

* пользователь выбирает опцию «изменение цвета фона»
* Пользователь выбирает цвет из предлагаемой палитры
* Пользователь нажимает стрелку «назад»
* Приложение сохраняет выбранный цвет в SharedPreferences
* После сохранения цвета главная страница использует выбранный цвет

5.2.9. Функция “виджеты”

Клиентское приложение

* Приложение обновляет “вопрос дня” (смотри: Функция “вопрос дня” - 5.2.1)
* Приложение отправляет запрос на обновление виджета через AppWidgetProvider
* AppWidgetProvider обрабатывает запрос
* AppWidgetProvider обновляет данные в виджете

  5.2.10. Функция “о приложении”

5.3. База данных.

 Структура базы данных

5.3.1. Таблица *questions*

Описание: хранит вопрос дня для функции “вопрос дня”

Поля:

* id: primary key, уникальный идентификатор вопроса
* question: текст вопроса

5.3.2. Таблица *text\_answers*

Описание: хранит текстовые ответ пользователей

Поля:

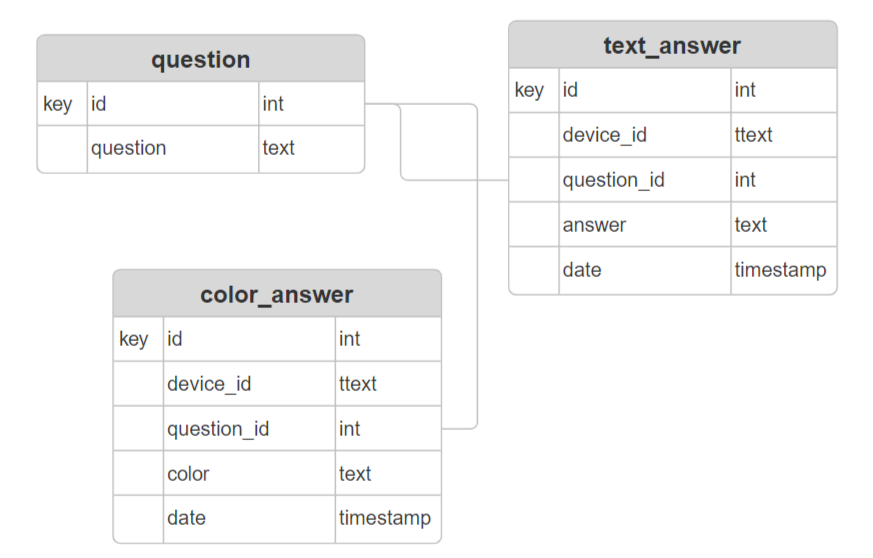
* id: Primary Key, уникальный идентификатор ответа
* device\_id: идентификатор устройства пользователя
* question\_id: внешний ключ, идентификатор вопроса
* answer: текст ответа
* date: дата и время ответа

5.3.3. Таблица color\_answers

Описание: хранит цветовые ответ пользователей.

Поля:

* id: Primary Key, уникальный идентификатор ответа
* device\_id: идентификатор устройства пользователя
* question\_id: внешний ключ, идентификатор вопроса
* color: выбранный цвет (HEX)
* date: дата и время ответа



*Рис. 3. ERD диаграмм сущностей*

**6. Интерфейсы пользователя**

* В приложение будет реализована стандартная система кнопок «назад», которая встроена в Android.
* Кнопка “назад” отвечает за сохранений изменений пользователя
* Принципом дизайна является минимализм и читаемость. Простая и интуитивно понятная навигация.
* Цветовая схема: стандартная цветовая схема начинается с бежевого цвета. Далее пользователь может выбрать цвет из палитры приложения.
* Визуальные элементы: кнопки, иконки и логотип.

6.1. Главный экран приложения

* Элементы:текстовое поле «вопрос дня»
* Кнопки: настройки, «ответ текстом», «ответ цветом», «Истрия ответов»

6.2. Экран “ответ тестом”

* Элементы: Поле для ввода текста, поле «вопрос дня»
* Кнопки: назад.

6.3. Экран “ответ цветом”

* Элементы: поле «вопрос дня», палитра цветов
* Кнопки: сохранить, назад.

6.4. Экран “история ответов”

* элементы оформлены списком: поле вопрос дня», поле «ответ», поле «дата ответа».
* Кнопки: назад.

6.5. Экран “поздсказка дня”

* элементы: поле с текстом
* Кнопки: назад.

6.6. Экран “настройки”.

* элементы: три поля с текстом-ссылкой
* Кнопки: назад.

 6.6.1. Экран “настройка частоты ответов”

* элементы: поле с тестом «настройка частоты ответа»
* Элементы: четыре кнопки с выбором ответа, назад.

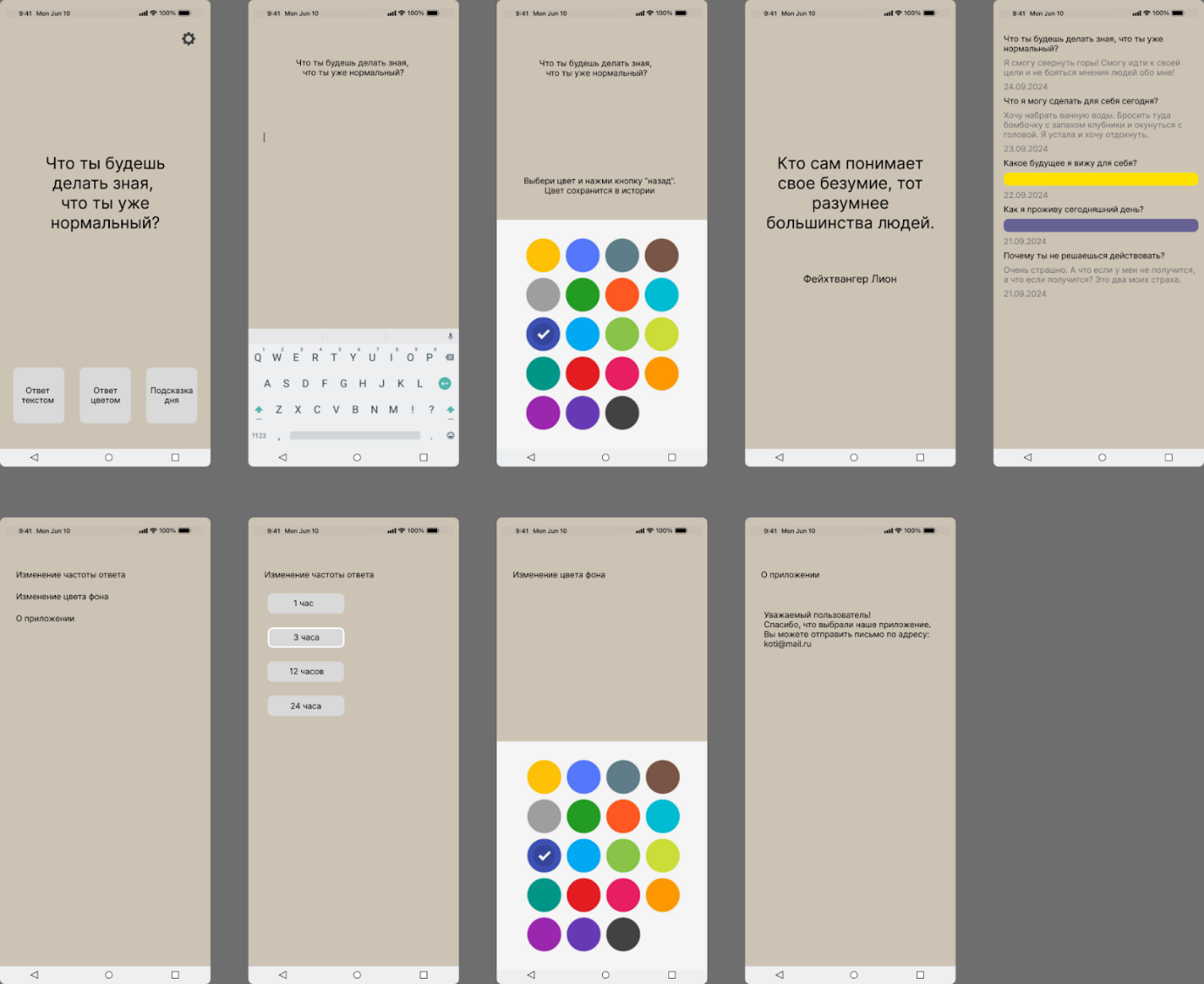
 6.6.2. Экран “изменение цвета фона”

* элементы: поле с тестом «изменение выбора цвета», палитра цветов
* Кнопки: назад.

 6.6.3. Экран «о приложении»

* элементы: поле с текстом
* Копки: назад

Макет экранов:



*Рис. 5. Прототип интерфейса*

**7. Внешнее API**

7.1. Описание ресурса

Quotes API предоставляет случайную цитату на русском языке для функции «подсказка дня».

Формат передачи данных: HTTP

Тип данных: JSON

7.2. Конечная точка и методы

7.2.1. Конечная точка: <https://nbvk-quotes-api.vercel.app/>

7.2.2. Методы:

GET /random - возвращает случайную цитату на русском языке

7.3. Пример запроса

*GET /random HTTP/1.1*

*Host: nbvk-quotes-api.vercel.app*

*Accept: application/json*

7.4. Пример ответа

*{*

*"quote": "Самое лучшее время посадить дерево было 20 лет назад. Второе лучшее время - сейчас.",*

*"author": "Китайская пословица"*

*}*

7.5. Коды состояния HTTP

* 200 OK: запрос выполнен успешно
* 404 NOT FOUND: цитата не найдена
* 500 Internal Server Error: внутренняя ошибка сервера

**8. Тестирование и валидация**

8.1. Тестирование

8.1.1. План тестирования

8.1.1.1. Функциональное тестирование

* Проверка корректности работы всех функций
* Тесты: вопрос дня, ответ текстом, ответ цветом, история ответов, подсказка дня, изменение цвета фона, настройка времени обновления вопроса, виджеты

8.1.1.2. Интеграционные тестирование

* Проверка корректного обмена данными между компонентами системы клиентским приложением, сервером и базой данных
* Проверка корректного обмена данными с внешними AP

8.1.1.3. Нагрузочного тестирование: проверка производительности и стабильности при большой нагрузке свыше 1000 запросов одновременно

8.1.1.4. Проверка безопасности: проверка защиты данных и безопасности передачи данных

8.1.2. Тест-кейсы:

8.1.2.1. «Получение вопроса дня»

Цель: проверить корректность получения «вопроса дня» и его обновление в пользовательском приложении

Предусловие: база данных должна содержать вопросы

Этапы:

* Отправить GET-запрос серверу на конечную точки */api/question*.
* Проверить, что статус код равен 200.
* Проверить, что в ответ содержится поле *question* и оно не пустое.
* Проверить, что вопрос действительно случайный.

Ожидаемый результат:

* вопрос дня успешно получен
* статус код равен 200.
* поле *question* присутствует и содержит текст
* при повторных запросах возвращаются разные вопросы

8.1.2.2. “Отправка текстового ответа пользователя”

Цель: проверка корректности отправки текстового ответа на сервер и сохранения в базе данных.

Предусловие:

* сервер работает и готов принимать запросы
* база данных настроена для хранения текстовых ответов.

Шаги:

* Отправить POST-запрос к серверу на конечную точку */api/saveAnswer* с телом, содержащим текстовый ответ
* Проверить, что статус код ответа равен 200.
* Проверить, что в ответе содержится поле *status* со значением *success*

Ожидаемый результат:

* текст ответа успешно отправлен
* статус код равен 200
* поле *status* равно *success*

8.1.2.3. “Отправка цветового ответа пользователя”

Цель: проверка корректности отправки цветового ответа и сохранение в базе данных

Предусловие:

* сервер работает и готов принимать запросы
* база данных настроена для хранения ответов

Шаги:

* Отправить POST-запрос к серверу на конечную точку */api/saveColorAnswer*
* Проверка, что статус код равен 200
* Проверить, что ответ содержит поле *status* со значением *success*

Ожидаемый результат:

* Цветовой ответ успешно отправлен
* Статус код равен 200
* Поле *status* равно *success*

8.1.2.4. “Получение истори  ответов”

Цель: проверка корректности получения истории ответов из баз данных

Предусловие:

* сервер работает и готов получать запросы
* База данных содержить историю ответов

Шаги:

* отправить GET-запрос к серверу на получение конечной точки */api/history* с параметром *deviceId*
* проверить, что статус код ответа равен 200
* Проверить, что в ответе содержится список ответов пользователя.

Ожидаемый результат:

* история ответов успешно получена
* статус код равен 200
* в ответе содержится список ответов пользователя

8.1.2.5. “Получение подсказки дня”

Цель: проверить корректность получения случайно цитаты дня от внешнего API

Предусловие:

* сервер работает и готов получать запросы
* внешний API доступен для запроса

Шаги:

* отправить GET-запрос к серверу на конечную точку */api/quote*
* Проверить, что статус код ответа равен 200
* Проверить, что в ответе содержится поле *quote* и оно не пустое
* Проверить, что в ответе содержится поле *author* и оно не пустое

Ожидаемый результат:

* цитат дня успешно получена
* статус код равен 200
* поле *quote* присутствует и содержит текст
* поле *author* присутствует и содержит текст

8.2. Валидация

Цель: подтверждение соответствия продукта требованиям и ожиданиям пользователей.

Основные вопросы валидации:

* соответствует ли приложение ТЗ?
* удовлетворяет ли приложение потребностям пользователей?
* предоставляет ли приложение ожидаемый функционал?

Процесс валидации:

* анализ требований: убедиться, что все функции приложения реализованы в соответствии с требованиями ТЗ
* пользовательское тестирование: провести тестирование с реальными пользователями, собрать обратную связь.
* проверка пользовательского интерфейса: оценить удобство, проверить соответствие дизайна и функциональных требований
* документирование результатов валидации: записать все результаты валидации и выявленных проблем, обозначить шаги по исправлению.

**9. Развертывание и эксплуатация**

Этапы развертывания

9.1. Подготовка окружения.

* установка необходимых инструментов и ПО (Node.js, PostgreSQL, Android Studio)
* настройка сервера для размещение бэкенда

9.2. Развертывание базы данных

* создание базы данных и необходимых таблиц
* наполнение базы данных

9.3. Развертывание сервера

* установить зависимости и настроить серверное приложение
* настроить окружение
* запустить сервер и протестировать его работу

9.4. Развертывание клиентского приложения

* установить зависимости и настроить проект в Android Studio
* настроить доступ к серверу
* Скомпилировать и протестировать приложение на различных этапах

9.5. Тестирование и отладка

* провести полное тестирование по тест-кейсам
* исправить найденные ошибки
* провести пользовательское тестирование

9.6. Публикация и развертывание

* опубликовать приложение в магазинах
* обеспечит круглосуточную доступность сервера  и базы данных для пользователей

9.7. Мониторинг и поддержка

Обеспечить поддержку и регулярное обновление приложения.

**10. Риски и ограничения.**

10.1. Риски:

10.1.1. Технические риски

* нестабильность серверной инфраструктуры: возможные сбои приводят к недоступности приложения
* безопасность данных: угрозы утечки данных
* интеграция с внешним API: внешнее API может быть недоступно или изменено без предупреждения, что может привести к сбоям в работе приложения

10.1.2. Операционные риски:

* ошибки в коде: непредвиденные баги, которые могут возникнуть в процессе разработки и эксплуатации.
* обновление и совместимость: риски, связанные с обновление операционных устройств пользовательских устройств, которые могут привести к несовместимости с приложением.

10.1.3. Бизнес риски:

* конкуренция: возможно появление аналогичных приложений на рынке, что может снизить интерес к приложению.

10.2. Ограничения:

10.2.1. Технические ограничения:

* ограничения платформ: возможности платформы Android и используемых технологий могут ограничить функциональность и производительность приложения
* ограничения внешних API: внешние API могут иметь ограниченное количество запросов, доступность и время отклика

10.2.2. Временные ограничения:

* сроки разработки: необходимость завершить разработку и тестирование в установленное время.

10.2.3. Ресурсные ограничения:

* бюджет: ограниченный бюджет на разработку, тестирование и маркетинг приложения.
* человеческие ресурсы: ограниченное количество и низкая квалификация специалистов по разработке и тестированию.

10.3. Меры по снижению рисков:

* использование надежного хостинга для серверной инфраструктуры
* подготовка резервных планов на случай недоступности внешних API
* гибкая архитектура, позволяющая вносить изменения, обновления, новые функции