Государственное бюджетное профессиональное образовательное учреждение

«Нижегородский Губернский колледж»

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

Информационной системы: «Code Blitz»

Разработчик\_\_\_\_\_\_\_\_\_Фадеев А. В. 01.10.2024

Заказчик\_\_\_\_\_\_\_\_\_\_\_\_Мамшева Ю. С. 01.10.2024

Нижний Новгород

2024

ОСНОВНАЯ ЧАСТЬ

# Общие сведения

## Наименование системы

Полное наименование системы – «Code Blitz».

Краткое наименование системы – «CBlitz».

# Цели и задачи создания системы

В современном мире программирование является одним из самых востребованных и сложных направлений профессиональной деятельности. Постоянная практика необходима для поддержания высокого уровня навыков, особенно в области алгоритмики, являющейся ключевым элементом разработки эффективных программных модулей. Однако, найти время и мотивацию для регулярных занятий бывает непросто.

Мобильное приложение, предоставляющее возможность обучаться написанию алгоритмов в игровой форме в любой момент времени, может стать решением этой проблемы. Доступность мобильных устройств в современном мире позволяет практиковаться в решении алгоритмических задач в любом месте и в любое время, делая процесс обучения более удобным и доступным.

Цель разработки – создание мобильного приложения, позволяющего пользователям решать ежедневные задачи по программированию в соревновательной форме, сравнивая свои результаты и решения с другими пользователями.

Для выполнения поставленной цели необходимо выполнение следующих задач:

* Разработать макет приложения;
* Разработать серверную часть приложения;
* Провести тестирование серверной части приложения;
* Разработать мобильное приложение;
* Провести тестирование мобильного приложения;
* Подготовить отчёт и презентацию о разработанном продукте;
* Презентовать разработанный продукт перед заказчиком.

# Выбор паттерна проектирования

Выбранный паттерн проектирования – MVVM (Model-View-ViewModel)

Обоснование выбора:

* Паттерн обеспечивает чёткое разделение интерфейса, бизнес логики и хранения данных, тем самым предавая упорядоченность структуре проекта и упрощая разработку;
* Разделение ответственности компонентов системы упрощает их тестирование по отдельности;
* Разделение компонентов делает приложение более масштабируемым и позволяет легче вносить в него изменения.

# Требования к серверной части

## Функциональные требования

Перечень функциональных требований к серверной части:

* Реализация на основе Supabase;
* Аутентификация пользователей: создание учётных записей, авторизация;
* Хранений задач: добавление задач в БД, хранение задач в БД, удаление задач из БД, хранение даты задачи (ввиду формата ежедневных задач);
* Хранение решений: добавление решений в БД, хранение решений в БД, хранение кода решения в БД, связность пользователя с решением в БД, хранение времени решения, связность решения и задачи, хранение данных о выбранном языке программирования, удаление решений;
* Наличие API для добавления и запроса данных из мобильного приложения (реализуется Supabase);

## Нефункциональные требования

Перечень нефункциональных требований к серверной части:

* Безопасность данных пользователя при аутентификации;
* Возможность расширения системы для обработки большего количества пользователей и задач.

# Требования к мобильному приложению

## Функциональные требования

Перечень функциональных требований к мобильному приложению:

* Регистрация и авторизация: создание учетной записи, вход в систему, выход из системы;
* Просмотр задач: получение ежедневных задач, просмотр описания и условий задачи;
* Реализация задач: выбор языка программирования, ввод кода, подсветка кода, запуск таймера, отправка решения на проверку;
* Просмотр результатов: просмотр таблицы лидеров, сравнение своего времени с результатами других пользователей;
* Просмотр решений: просмотр решений других пользователей, возможность изучения их кода.

## Нефункциональные требования

Перечень нефункциональных требований к мобильному приложению:

* Пользовательский интерфейс: простота использования, интуитивно понятный дизайн, адаптируемый для различных размеров экрана;
* Быстрая загрузка приложения, быстрое выполнение операций.

# Выбор ПО для реализации

## Мобильное приложение

Для реализации мобильного приложения было выбрано следующее ПО:

1. Среда разработки Android Studio:

* Официальная IDE для Android: Android Studio предоставляет богатый набор инструментов для разработки приложений под Android, включая симуляторы, отладчик, средства профилирования и многое другое;
* Большая экосистема: Android Studio поддерживается Google и имеет большую экосистему плагинов и документации, что облегчает разработку.

1. Фреймворк Jetpack Compose:

* Декларативный подход: Jetpack Compose позволяет создавать пользовательские интерфейсы декларативно, описывая структуру UI с помощью функций, что делает код более читаемым и легко изменяемым.
* Создание современных UI: Jetpack Compose предназначен для создания современных и динамичных UI с поддержкой анимаций и переходов.
* Простой в изучении: Jetpack Compose относительно прост в изучении, что сокращает время разработки и позволяет быстро приступить к работе.

1. Figma:

* Figma доступна в веб-браузере, что позволяет работать с ней на любом устройстве без необходимости установки;
* Figma предлагает широкий набор инструментов для создания высококачественных дизайнов, включая векторную графику, прототипирование, стили, компоненты и анимации.

## Серверная часть

Для реализации серверной части был выбран сервис Supabase:

* Сервис предоставляет полноценную платформу, включая базу данных (PostgreSQL), аутентификацию, хранилище файлов и API;
* Supabase обладает простым в использовании интерфейсом и предлагает богатую документацию, что сокращает время на настройку и разработку серверной части;
* Supabase является масштабируемым решением, что позволяет приложению обрабатывать большое количество запросов и данных;
* Supabase имеет бесплатный план, что делает его доступным для небольших и учебных проектов.