Государственное бюджетное профессиональное образовательное учреждение

«Нижегородский Губернский колледж»

**Итоговый отчёт**

По разработке Информационной системы: «Code Blitz»

Студент\_\_\_\_\_\_\_\_\_Фадеев А. В. 01.10.2024

Преподаватель\_\_\_\_\_\_\_\_\_\_\_\_Мамшева Ю. С. 01.10.2024

Нижний Новгород

2024

ВВЕДЕНИЕ

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 техническое задание;
* 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.

# ОСНОВНАЯ ЧАСТЬ

# Разработка технического задания

# 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. 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.

## Разработка макета приложения

## Выбор цветовой палитры

В рамках разработки макета приложения, был выполнен выбор основной цветовой палитры, используемой приложением:

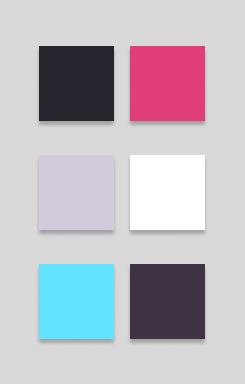


Рисунок 1 – Основная палитра приложения

Также был добавлен ряд дополнительных палитр, которые пользователи смогут устанавливать во время своей работы:



Рисунок 2 – Дополнительные палитры

## Выбор шрифта для заголовков и основного текста

Были выбраны следующие шрифты:

* Основной текст – JetBrains Mono
* Заголовки – Intro

## Разработка макета каждого экрана

Был разработан макет для каждого экрана приложения CodeBlitZ

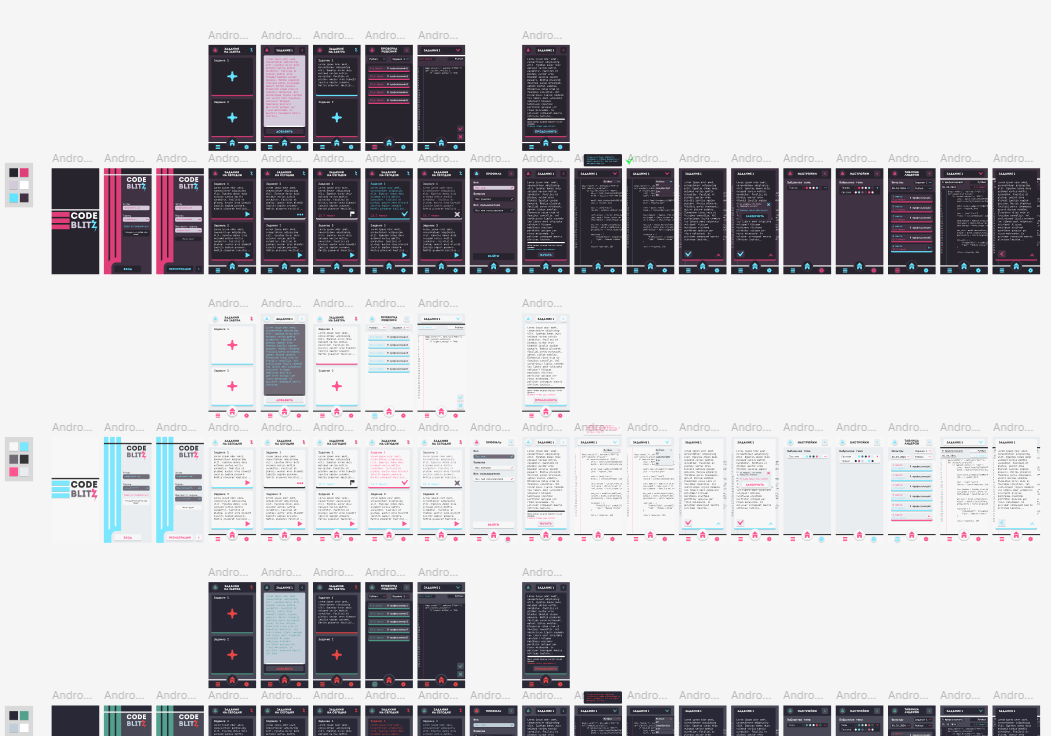


Рисунок 3 – Разработанный макет

Ссылка на макет - <https://www.figma.com/design/rTUUWG5ZbsedkDY6VORlQI/CodeBlitZ?node-id=0-1&t=pJLvMVX7pgUbVamx-1>

## Разработка серверной части приложения

## Проектирование серверной части

В качестве платформы реализации серверной части был выбран Supabase. Supabase предоставляет все необходимые инструменты для создания базы данных, аутентификации, хранилища объектов, функций и т.д.

Для серверной части была создана ER-диаграмма:

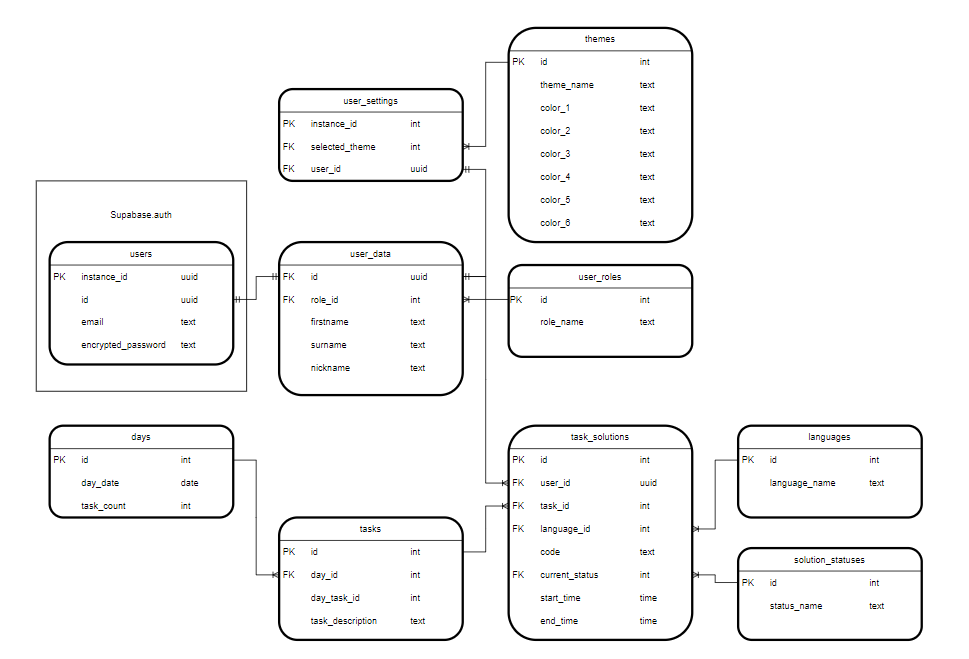


Рисунок 4 – ER-диаграмма

Всего в серверной части используется 10 таблиц:

Таблица 1 – Описание таблиц

| Таблица | Описание |
| --- | --- |
| users | Содержит информацию о пользователях (часть Supabase.auth) |
| user\_data | Содержит расширенные данные о пользователях, не относящиеся к аутентификации |
| user\_roles | Содержит информацию о ролях пользователей |
| user\_settings | Содержит информацию о настройках пользователя |
| days | Содержит информацию о днях с задачами |
| tasks | Содержит информацию о задачах |
| task\_solutions | Содержит информацию о решениях задач пользователей |
| languages | Содержит информацию о доступных языках программирования |
| solution\_statuses | Содержит информацию о статусах решения задачи |
| themes | Содержит информацию о доступных темах |

Далее описаны все поля таблиц:

Таблица 2 – Описание полей

| Таблица | Поле | Назначение | Тип данных | Ограничения |
| --- | --- | --- | --- | --- |
| users | instance\_id | Идентификатор записи о пользователе | uuid | primary key |
|  | id | Идентификатор пользователя | uuid | unique |
|  | email | Электронный адрес пользователя | text | not null |
|  | encrypted\_password | Зашифрованный пароль пользователя | text | not null |
| user\_data | id | Идентификатор пользователя | uuid | primary key |
|  | firstname | Имя пользователя | text | not null, length < 30 |
|  | surname | Фамилия пользователя | text | not null, length < 30 |
|  | role\_id | Идентификатор роли пользователя | integer | foreign key |
|  | nickname | Никнейм пользователя | text | length < 30 |
| user\_roles | id | Идентификатор роли | integer | primary key |
|  | role\_name | Название роли | text | not null, unique |
| user\_settings | id | Идентификатор настройки | integer | primary key |
|  | selected\_theme | Выбранная пользователем тема | integer | foreign key,not null |
|  | user\_id | Идентификатор пользователя | uuid | foreign key, not null, unique |
| days | id | Идентификатор дня | integer | primary key |
|  | day\_date | Дата дня | date | not null, unique |
|  | task\_count | Количество задач в дне | integer | not null, 0 <= task\_count <= 2 |
| tasks | id | Идентификатор задачи | integer | primary key, not null |
|  | day\_id | Идентификатор дня, к которому относится задача | integer | foreign key, not null |
|  | day\_task\_id | Номер задачи в рамках дня | integer | not null, 1 <= day\_task\_id <= 2 |
|  | task\_description | Описание задачи | text | not null, unique |
| task\_solutions | id | Идентификатор решения задачи | integer | primary key |
|  | user\_id | Идентификатор пользователя, решившего задачу | uuid | foreign key, not null |
|  | task\_id | Идентификатор задачи | integer | foreign key, not null |
|  | language\_id | Идентификатор языка, на котором решена задача | integer | foreign key, not null |
|  | code | Решение задачи | text | not null |
|  | current\_status | Текущий статус решения задачи | text | not null |
|  | start\_time | Время начала решения задачи | time | not null |
|  | end\_time | Время окончания решения задачи | time | not null |
| languages | id | Идентификатор языка | integer | primary key |
|  | language\_name | Название языка | text | not null, unique |
| solution\_statuses | id | Идентификатор статуса решения задачи | integer | primary key |
|  | status\_name | Название статуса решения задачи | text | not null, unique |
| themes | id | Идентификатор темы | integer | primary key |
|  | theme\_name | Название темы | text | not null |
|  | color\_1 | Цвет 1 для темы | text |  |
|  | color\_2 | Цвет 2 для темы | text |  |
|  | color\_3 | Цвет 3 для темы | text |  |
|  | color\_4 | Цвет 4 для темы | text |  |
|  | color\_5 | Цвет 5 для темы | text |  |
|  | color\_6 | Цвет 6 для темы | text |  |

Далее описаны все связи между таблицами:

Таблица 3 – Связи

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 1 | Таблица 2 | Тип связи | Описание |
| users | user\_data | 1-1 | Каждый пользователь имеет один набор данных |
| user\_data | user\_settings | 1-1 | Каждый пользователь имеет одну настройку |
| user\_data | user\_roles | м-1 | Пользователь может иметь только одну роль |
| user\_data | task\_solutions | 1-м | Пользователь может иметь множество решений |
| days | tasks | 1-м | День может содержать несколько задач |
| tasks | task\_solutions | 1-м | Задача может иметь много решений |
| user\_settings | themes | м-1 | Пользователь может выбрать одну тему |
| task\_solutions | languages | м-1 | На каждом языке программирования может быть написано несколько решений |
| task\_solutions | solution\_statuses | м-1 | Каждое решение задачи может иметь один из статусов |

Также была реализована функция Realtime для таблиц task\_solutions и user\_settings, однако в дальнейшем не была использована ввиду ограничений данного функции.

Всего для серверной части было разработано 3 триггера:

Таблица 4 – Триггеры

|  |  |
| --- | --- |
| Название | Описание |
| check\_task\_count\_trigger | Запрещает добавление задание на день, если для этого дня уже есть 2 задания |
| increment\_task\_count\_trigger | Увеличивает счетчик заданий на день после добавлении задания |
| set\_day\_task\_id\_trigger | Устанавливает day\_task\_id задачи в рамках дня (1 или 2) |

Листинг 1 – Триггер check\_task\_count\_trigger

|  |
| --- |
| *-- Триггер для проверки количества задач на день*  **CREATE** **OR** **REPLACE** **FUNCTION** check\_task\_count() **RETURNS** **TRIGGER** **AS** $$  **BEGIN**  **IF** (**SELECT** **COUNT**(\*) **FROM** tasks **WHERE** day\_id = NEW.day\_id) >= 2 **THEN**  **RAISE** **EXCEPTION** 'Нельзя добавить больше двух задач на этот день.';  **END** **IF**;  RETURN NEW;  **END**;  $$ LANGUAGE plpgsql;  **CREATE** **TRIGGER** check\_task\_count\_trigger  **BEFORE** **INSERT** **ON** tasks  **FOR** **EACH** **ROW**  **EXECUTE** **PROCEDURE** check\_task\_count(); |

Листинг 2 – Триггер increment\_task\_count\_trigger

|  |
| --- |
| *-- Триггер для увеличения количества задач на день*  **CREATE** **OR** **REPLACE** **FUNCTION** increment\_task\_count() **RETURNS** **TRIGGER** **AS** $$  **BEGIN**  **UPDATE** **days** **SET** task\_count = task\_count + 1 **WHERE** **id** = NEW.day\_id;  RETURN NEW;  **END**;  $$ LANGUAGE plpgsql;  **CREATE** **TRIGGER** increment\_task\_count\_trigger  **AFTER** **INSERT** **ON** tasks  **FOR** **EACH** **ROW**  **EXECUTE** **PROCEDURE** increment\_task\_count(); |

Листинг 3 – Триггер set\_day\_task\_id\_trigger

|  |
| --- |
| *-- Триггер для установки day\_task\_id*  **CREATE** **OR** **REPLACE** **FUNCTION** set\_day\_task\_id() **RETURNS** **TRIGGER** **AS** $$  **BEGIN**  **IF** (**SELECT** **COUNT**(\*) **FROM** tasks **WHERE** day\_id = NEW.day\_id) = 0 **THEN**  NEW.day\_task\_id := 1;  ELSE  NEW.day\_task\_id := 2;  **END** **IF**;  RETURN NEW;  **END**;  $$ LANGUAGE plpgsql;  **CREATE** **TRIGGER** set\_day\_task\_id\_trigger  **BEFORE** **INSERT** **ON** tasks  **FOR** **EACH** **ROW**  **EXECUTE** **PROCEDURE** set\_day\_task\_id(); |

## Разработка серверной части

Серверная часть была разработана в сервисе Supabase. Были реализованы все спроектированные элементы.

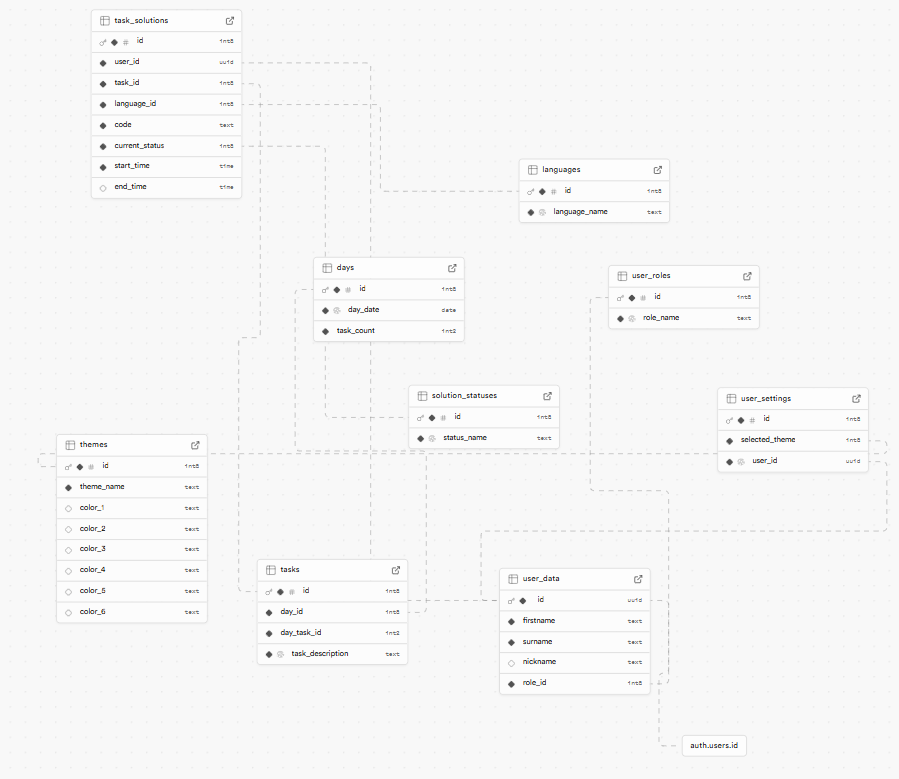


Рисунок 5 – Схема в Supabase

## Тестирование серверной части

## Разработка документации для тестирования сервера приложения

Для тестирования серверной части приложения была разработана следующая документация:

1. Тест-план;
2. Тестовый набор;
3. Тест-кейсы.

## Автоматизированное тестирование серверной части

Всего для серверной части было разработано 10 Юнит-тестов. Все юнит-тесты были написаны на языке SQL и выполнены с помощью установленного Supabase CLI.

Таблица 5 – Юнит-тесты

| Номер | Название | Описание | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Проверка, что в схеме public присутствуют все 9 таблиц | Проверяет число таблиц в схеме Public | Число таблиц - 9 |
| 2 | Проверка, что в счётчике заданий на текущий день значение равно 2 | После вставки нового дня в базу и добавления к нему двух заданий проверяет счетчик заданий у данного дня | Значение счетчика - 2 |
| 3 | Проверка, что в таблицу заданий за текущий день добавилось 2 задания | После вставки нового дня в базу и добавления к нему двух заданий проверяет количество заданий в таблице с заданиями за текущий день | Число заданий - 2 |
| 4 | Проверка, что первое добавленное задание имеет id в рамках дня 1 | После вставки нового дня в базу и добавления к нему двух заданий проверяет индекс первого задания в рамках дня | Индекс - 1 |
| 5 | Проверка, что второе добавленное задание имеет id в рамках дня 2 | После вставки нового дня в базу и добавления к нему двух заданий проверяет индекс второго задания в рамках дня | Индекс - 2 |
| 6 | Проверка, что при попытке добавить третье задание на день происходит исключение | Проверяет, происходит ли исключение при попытке добавить задание на день, у которого уже есть 2 задания | Исключение «Нельзя добавить больше двух задач на этот день.» |
| 7 | Проверка ограничения уникальности у ролей | Проверяет, что нельзя добавить роль, если в БД уже существует роль с таким названием | Исключение «duplicate key value violates unique constraint» |
| 8 | Проверка значения по умолчанию у темы приложения в настройках | Проверяет значение темы по умолчанию при добавлении записи с настройками пользователя | Индекс темы - 1 |
| 9 | Проверка ограничения уникальности у настроек пользователя | Проверяет, что для каждого пользователя возможна только одна запись с настройками | Исключение «duplicate key value violates unique constraint» |
| 10 | Проверка каскадного удаления у параметров пользователя | Проверяет, что при удалении пользователя удаляется запись о его настройках | Запись удалена |

Ниже представлен листинг всех юнит-тестов:

Листинг 4 – Юнит-тесы

|  |
| --- |
| begin;  select plan( 10 );  --Юнит-тест 1: Проверка, что в схеме public присутствуют все 9 таблиц  SELECT is(cast(count(\*) as integer), 9) FROM information\_schema.tables  WHERE table\_schema = 'public';  --Вставка данных о дне и задачах на день для тестов ниже  BEGIN;  INSERT INTO days(day\_date) VALUES (CURRENT\_DATE);  INSERT INTO tasks(day\_id, task\_description) VALUES ((SELECT id FROM days WHERE day\_date = CURRENT\_DATE), 'test1');  INSERT INTO tasks(day\_id, task\_description) VALUES ((SELECT id FROM days WHERE day\_date = CURRENT\_DATE), 'test2');  COMMIT;  --Юнит-тест 2: Проверка, что в счётчике заданий на текущий день значение равно 2  SELECT is(cast((SELECT task\_count FROM days WHERE day\_date = CURRENT\_DATE)as integer), 2);  --Юнит-тест 3: Проверка, что в таблицу заданий за текущий день добавилось 2 задания  SELECT is(cast(count(\*) as integer), 2) from tasks WHERE day\_id = (SELECT id FROM days WHERE day\_date = CURRENT\_DATE);  --Юнит-тест 4 и 5: Проверка, что добавленные задания имеют id в рамках дня 1 и 2  SELECT is(cast(day\_task\_id as integer), 1) FROM tasks WHERE task\_description = 'test1';  SELECT is(cast(day\_task\_id as integer), 2) FROM tasks WHERE task\_description = 'test2';  --Юнит-тест 6: Проверка, что при попытке добавить третье задание на день происходит исключение  SELECT throws\_like('INSERT INTO tasks (day\_id, task\_description) VALUES ((SELECT id FROM days WHERE day\_date = CURRENT\_DATE),''3131'')', 'Нельзя добавить больше двух задач на этот день.');  --Удаление записей для очистки БД от данных, используемых для тестирования  BEGIN;  DELETE FROM tasks WHERE day\_id = (SELECT id FROM days WHERE day\_date = CURRENT\_DATE);  DELETE FROM days WHERE id = (SELECT id FROM days WHERE day\_date = CURRENT\_DATE);  COMMIT;  --Вставка данных о пользователе и его настройках для тестов ниже  BEGIN;  TRUNCATE TABLE themes RESTART IDENTITY CASCADE;  INSERT INTO user\_roles(role\_name) VALUES ('test');  INSERT INTO user\_data(id, firstname, surname, nickname, role\_id) VALUES (CAST('3818e8d2-79e3-40f8-9790-f10ecd332554' AS UUID), 'test', 'test', 'test', (SELECT id from user\_roles where role\_name = 'test'));  INSERT INTO themes(theme\_name) VALUES('test');  INSERT INTO user\_settings(user\_id) VALUES (CAST('3818e8d2-79e3-40f8-9790-f10ecd332554' AS UUID));  COMMIT;  --Юнит-тест 7: Проверка ограничения уникальности у ролей  SELECT throws\_like('INSERT INTO user\_roles(role\_name) VALUES (''test'');', 'duplicate key value violates unique constraint%');  --Юнит-тест 8: Проверка значения по умолчанию у темы приложения в настройках  SELECT is(cast(selected\_theme as integer), 1) FROM user\_settings WHERE user\_id = CAST('3818e8d2-79e3-40f8-9790-f10ecd332554' AS UUID);  --Юнит-тест 9: Проверка ограничения уникальности у настроек пользователя  SELECT throws\_like('INSERT INTO user\_settings(user\_id) VALUES (CAST(''3818e8d2-79e3-40f8-9790-f10ecd332554'' AS UUID));', 'duplicate key value violates unique constraint%');  --Юнит-тест 10: Проверка каскадного удаления у параметров пользователя  BEGIN;  DELETE FROM user\_data where id = CAST('3818e8d2-79e3-40f8-9790-f10ecd332554' AS UUID);  DELETE FROM user\_roles where role\_name = 'test';  DELETE FROM themes where theme\_name = 'test';  COMMIT;  SELECT is(cast(count(\*) as integer), 0) FROM user\_settings WHERE user\_id = CAST('3818e8d2-79e3-40f8-9790-f10ecd332554' AS UUID);  select \* from finish();  rollback; |

Все юнит-тесты были успешно пройдены:

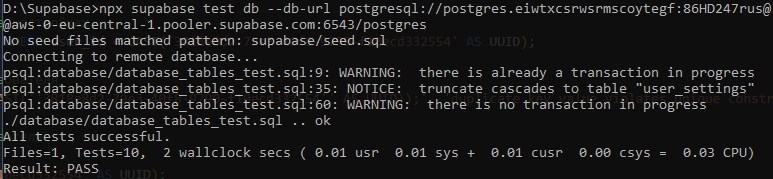


Рисунок 6 – Выполнение юнит-тестов

## Разработка мобильного приложения

## Подключение шрифтов и настройка ранее выбранных цветов

В приложении были подключены используемые шрифты JetBrains-Mono и Intro.

Также в приложении были прописаны цвета, которые относятся к его основной теме. Остальные цвета приложение получает из базы данных.

Листинг 5 – Подключение шрифтов

|  |
| --- |
| //Используемые шрифты val *Intro* = *FontFamily*(  *Font*(R.font.*intro*, FontWeight.Normal) )  val *JetBrains* = *FontFamily*(  *Font*(R.font.*jet\_brains\_mono\_regular*, FontWeight.Normal),  *Font*(R.font.*jet\_brains\_mono\_bold*, FontWeight.Bold) ) |

Листинг 6 – Цвета основной темы

|  |
| --- |
| //Базовые цвета приложения при запуске val *Background* = *Color*(0xff28242D) val *Primary* = *Color*(0xffDF3E78) val *SecondaryContainer* = *Color*(0xffD3CADC) val *Tertiary* = *Color*(0xffFFFFFF) val *Secondary* = *Color*(0xff62E3FF) val *OnBackground* = *Color*(0xff3F3343) |

## Верстка макета

Была выполнена верстка всех экранов, созданных в макете:



Рисунок 7 – Экран авторизации и регистрации