Содержание

Введение

По предмету ТРПО была поставлена задача, разработать мобильное приложение «StudyPal».

Цель проекта заключается в создании приложения «StudyPal». Главной задачей при автоматизации этого проекта является разработка мобильного приложения «StudyPal».

Создаваемый программный продукт будет предоставлять возможность просмотра заданий и помощи по ним.

Далее приведём краткое описание разделов пояснительной записки.

Первый раздел носит название «Анализ задачи». В данном разделе будут представлены: постановка задачи, диаграмма вариантов использования, модель данных, техническое задание. Во втором разделе «Проектирование задачи» рассмотрены основные аспекты разработки программного продукта. Будет присутствовать описание выбранной модели жизненного цикла, инструменты разработки, UMLдиаграммы, пользовательский интерфейс, тесты на использование, а так же план над проектом. «Реализация» – это третий раздел пояснительной записки, в котором будет описана структура базы данных т руководство программиста. «Тестирование» – это четвёртый раздел пояснительной записки. В этом разделе будет описано функциональное тестирование данного проекта. Также будет предоставлен вывод о результатах тестирования. Пятый раздел – «Оценка качества». Данный раздел будет содержать характеристики проекта, сортируемые по критериям оценки качества. Шестой раздел – «Руководство пользователя». В данном разделе будут описаны общие сведения о проекте и процессы его эксплуатации. «Заключение» будет содержать краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств, а также заключительный вывод о проделанной работе.

В «Списке используемых источников» будут содержаться различные источники, которыми разработчик пользовался на протяжении всего цикла разработки программного продукта.

1 Анализ задачи

1.1 Постановка задачи

1.1.1 Организационно-экономическая сущность задачи

Темой данного проекта является разработка мобильного приложения «StudyPal». Оно предоставляет возможность выполнения учебных заданий и помощи студентам при их выполнении.

Целью разработки является создание мобильного приложения выполнения студентами заданий и, если нужно, просмотра помощи по ним. В поставленной задаче необходимо реализовать простой и приятный интерфейс, позволяющий использовать проект пользователю, не обладающему дополнительными знаниями ЭВМ.

На данный момент существует такое популярное приложение, как Google Class. В этом приложении есть курсы, слушателем которых пользователь является. Также в этом приложении есть кнопка выбора аккаунта, есть меню, ещё есть кнопка “+”, с помощью которой пользователь может добавиться в новый курс.

1.1.2 Функциональные требования

К поставленной задаче были заявлены следующие функциональные требования, которые сможет выполнить пользователь:

- Работа в личном кабинете
- Просмотр помощи по заданиям
- Авторизация
- Просмотр заданий

Функциональные требования, которые сможет выполнить гость:

- Регистрация

1.1.3 Описание входной, выходной и условно-постоянной информации

Таблица 1 – Описание информации

№	Категория пользователей	Наименование процесса	Краткое описание алгоритма выполнения процесса	Входная информация	Выходная информация	Условно-постоянная информация
1	2	3	4	5	6	7
2	Гость	Регистрация	При нажатии будет окно регистрации (или входа для зарегистрированного пользователя), нужно заполнить требуемы поля и после этого гость получит больше функций в этом приложении	Заполнение полей регистрации (имя, электроннная почта/номер телефона, пароль)	Вход в личный кабинет	Отсутствует
3	Пользователь	Работа в личном кабинете	Просмотр выполненных заданий, редактирование профиля	Отсутствует	Выполненные задания	Отсутствует
4	Пользователь	Просмотр помощи по заданиям	При нажатии будет экран, на котором будут расположены изображения, на которых изображён процесс выполнения конкретной части задания	Отсутствует	Изображения с помощью по заданиям	Отсутствует
5	Пользователь	Авторизация		Логин и пароль	Открытие личного кабинета	
6	Пользователь	Просмотр заданий	Пользователь может просматривать задания, выложенные администратором	Отсутствует	Задания	

1.1.4 Нефункциональные требования

Требования к применению: Показывает задания на выполнение и помощь к ним.

Требования к реализации: Приложение должно быть написано на языке Dart в фреймворке Flutter, и должно быть совместимо с операционной системой Android.

Требования к надёжности: Система может быть недоступна не более чем 24 часа в год.

Требования к интерфейсу: При разработке приложения должны быть использованы преимущественного белые/фиолетовые оттенки. Основные разделы приложения должны быть доступны с первой страницы. Грамотный пользовательский интерфейс.

1.2 Диаграмма вариантов использования

Диаграмма вариантов использования – диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актёров, взаимоействующих с системой с помощью так называемых вариантов использования.

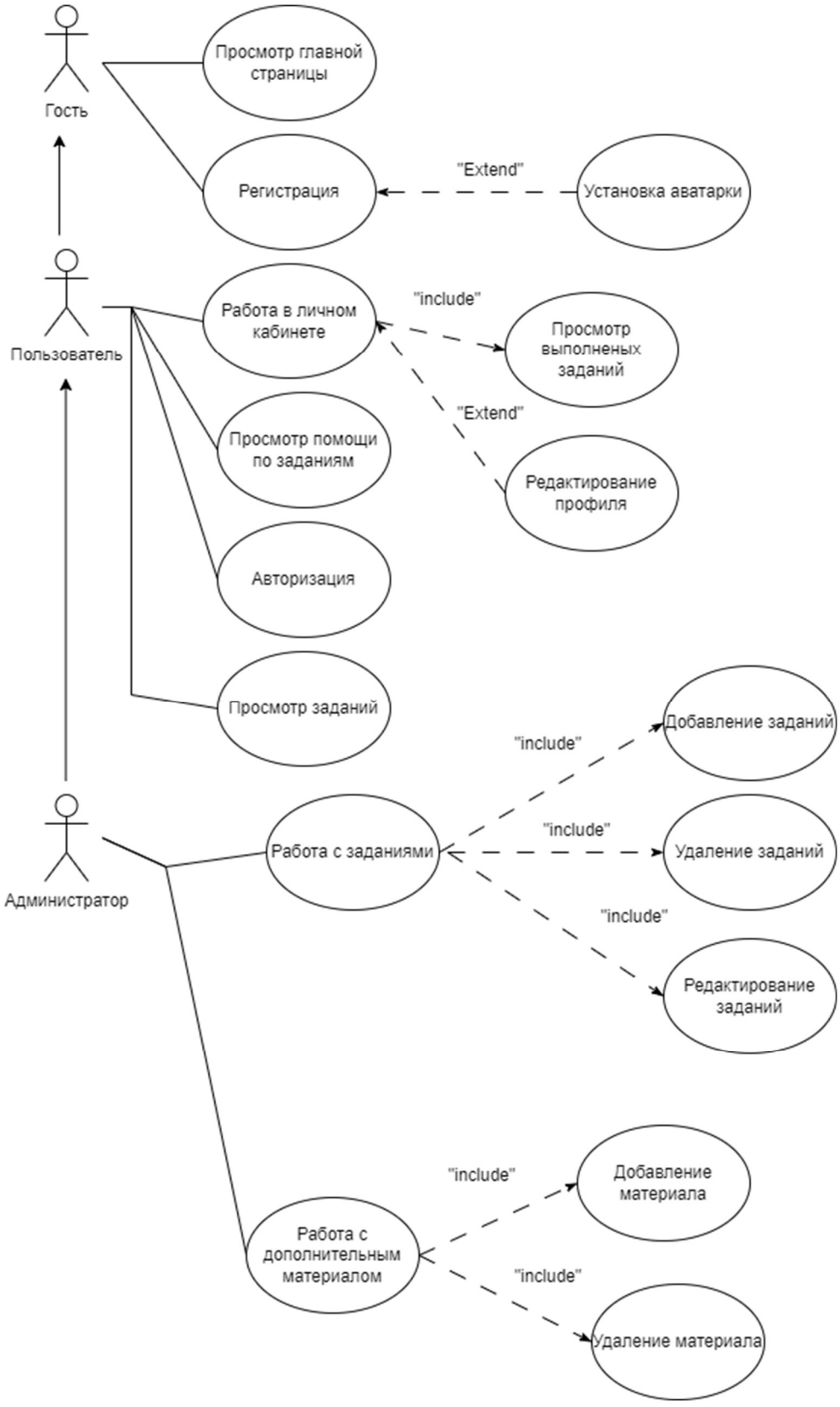

Рисунок 1 – Графической изображение диаграммы вариантов использования

1.3 Инструменты разработки

Для разработки данного проекта будет выбрана среда разработки Visual Studio Code, которая является наиболее актуальной средой для создания приложений данного типа.

Разработка будет производится на языке программирования Dart – для разработки мобильных приложений, современных веб-приложений, настольных приложений и интернета вещей с использованием фреймворка Flutter. Он также поддерживает несколько продвинутых концепций, таких как интерфейсы, примеси, абстрактные классы, уточнённые обобщения и типовые интерфейсы.

Мобильные прилжения и веб-приложения Google – два из самых распространённых применений языка Dart. Dart также позволяет компилировать приложения в нативный машинный код для мгновенного запуска.

Иные инструменты, используемые для разработки и написании сопутствующей документации:

- WEB-ресурс DRAW.IO – будет использоваться для создания графической части и разработки UML-диаграмм;
- Microsoft Office Word – для написания документации к программному продукту.
- Microsoft Access – для создания модели базы данных.
- Figma – для создания прототипа интерфейса.

Разработка проекта будет происходить на компьютере со следующими параметрами:

- процессор Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz;
- объём оперативной памяти 8.00 GB;
- объём места на SSD диске 250 GB;
- видеокарта Intel(R) UHD Graphics;
- ОС Window 10 Home;

1.4 Выбор стратегии разработки и модели ЖЦ

Таблица 2 – выбор модели жизненного цикла на основе характеристик команды разработчиков.

№ критерия	Критерии категории требований	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
1.	Являются ли требования к проекту легко определимыми и реализуемыми?	Да	Да	Нет	Нет	Нет	Да
2.	Могут ли требования быть сформулированы в начале ЖЦ?	Нет	Да	Да	Да	Нет	Нет
3.	Нужно ли демонстрировать требование с целью их определения?	Нет	Да	Да	Да	Да	Нет
4.	Требуется ли проверка концепции программного средства или системы?	Да	Нет	Нет	Нет	Нет	Да
5.	Требуется ли проверка концепции программного средства или системы?	Нет	Да	Нет	Да	Да	Нет
6.	Будут ли требования изменяться или уточняться с ростом сложности системы (программного средства) в ЖЦ?	Да	Да	Нет	Да	Нет	Нет
7.	Нужно ли реализовать основные требования на ранних этапах разработки?	Да	Нет	Да	Нет	Нет	Нет
	Итог	4	5	3	4	2	2

Таблица 3 – Выбор модели жизненного цикла на основе характеристик команды разработчиков.

№ критерия	Критерии категории команды разработчиков проекта	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
1.	Являются ли проблемы предметной области проекта новыми для большинства разработчиков?	Нет	Нет	Да	Да	Нет	Нет
2.	Являются ли инструментальные средства, используемые в проекте, новыми для большинства разработчиков?	Да	Да	Да	Нет	Да	Нет
3.	Изменяются ли роли участников проекта на протяжении ЖЦ?	Да	Нет	Нет	Нет	Да	Нет
4.	Является ли структура процесса разработки более значимой для разработчиков, чем гибкость?	Нет	Да	Нет	Да	Да	Нет
5.	Важна ли легкость распределения человеческих ресурсов проекта?	Да	Да	Нет	Нет	Нет	Да
6.	Приемлет ли команда разработчиков оценки, проверки, стадии разработки?	Да	Нет	Да	Не	Нет	Нет
	Итог	4	3	3	2	3	1

Таблица 4 – Выбор модели жизненного цикла на основе характеристик коллектива пользователей.

№ критерия	Критерии категории коллектива пользователей	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
1.	Будет ли присутствие пользователей ограничено в ЖЦ разработки?	Да	Да	Нет	Нет	Да	Нет
2.	Будут ли пользователи оценивать текущее состояние программного продукта (системы) в процессе разработки?	Да	Нет	Нет	Да	Нет	Нет
3.	Будут ли пользователи вовлечены во все фазы ЖЦ разработки?	Нет	Нет	Да	Да	Да	Нет
4.	Будет ли заказчик отслеживать ход выполнения проекта?	Да	Нет	Нет	Да	Да	Да
	Итог	3	1	1	3	3	1

Таблица 5 – Выбор модели жизненного цикла на основе характеристик типа проектов и рисков.

№ критерия	Критерии категории типов проекта и рисков	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирован ия	Эволюционная
1.	Разрабатывается ли в проекте продукт нового для организации направления?	Нет	Нет	Да	Нет	Да	Да
2.	Будет ли проект являться расширением существующей системы?	Да	Нет	Нет	Да	Да	Нет
3.	Будет ли проект крупно- или среднемасштабным?	Да	Да	Нет	Нет	Нет	Да
4.	Ожидается ли длительная эксплуатация продукта?	Нет	Да	Нет	Да	Да	Нет
5.	Необходим ли высокий уровень надёжности продукта проекта?	Нет	Да	Да	Да	Нет	Нет
6.	Предполагается ли эволюция продукта проекта в течение ЖЦ?	Да	Нет	Да	Да	Да	Нет

Продолжение таблицы 5

7.	Велика ли вероятность изменения системы (продукта) на	Да	Нет	Нет	Нет	Да	Да
8.	Является ли график сжатым?	Да	Да	Да	Да	Нет	Да
9.	Предполагается ли повторное использование компонентов?	Да	Нет	Нет	Нет	Да	Да
10.	Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)?	Нет	Нет	Да	Да	Нет	Нет
	Итог	6	4	5	5	6	5

№ Критерия	Итог	17	13	12	14	14	9

2 Проектирование

2.1 Проектирование системы меню

Рисунок 2 – Графическое изображение системы главного меню

2.2 Модель данных

Рисунок 3 – Графическое изображение модели данных

2.3 Моделирование бизнес-процессов

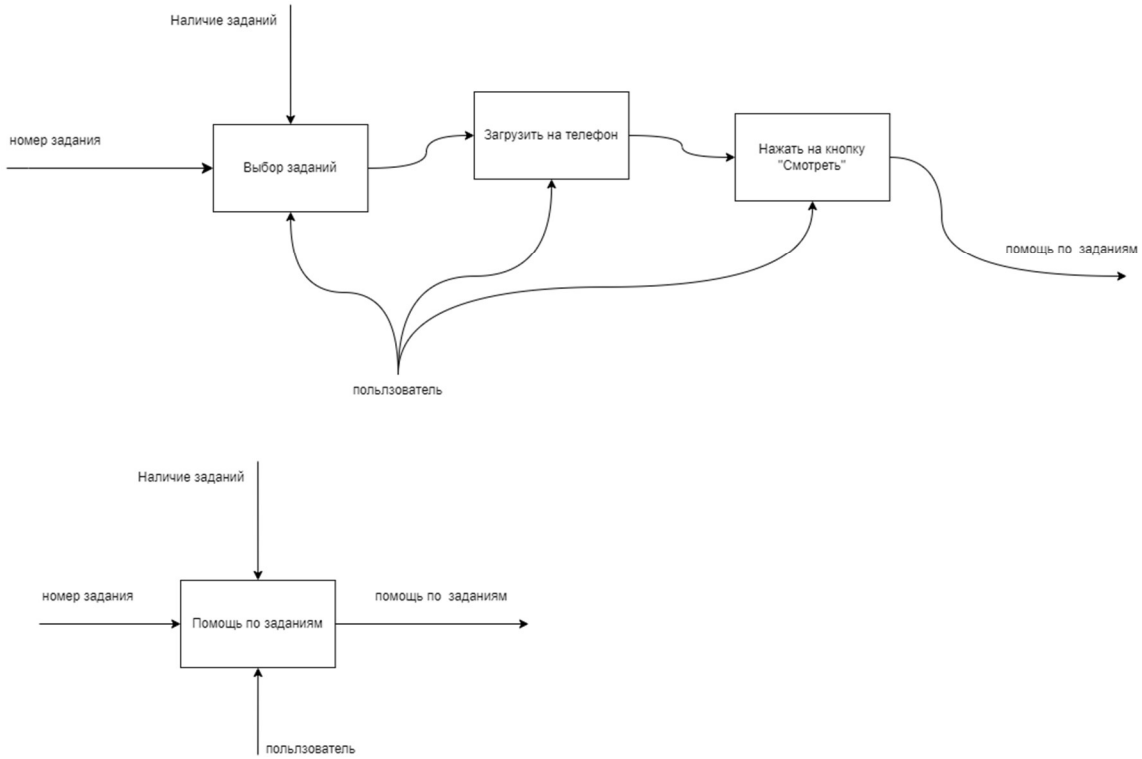

Рисунок 3 – Графическое изображени функциональной модели процесса

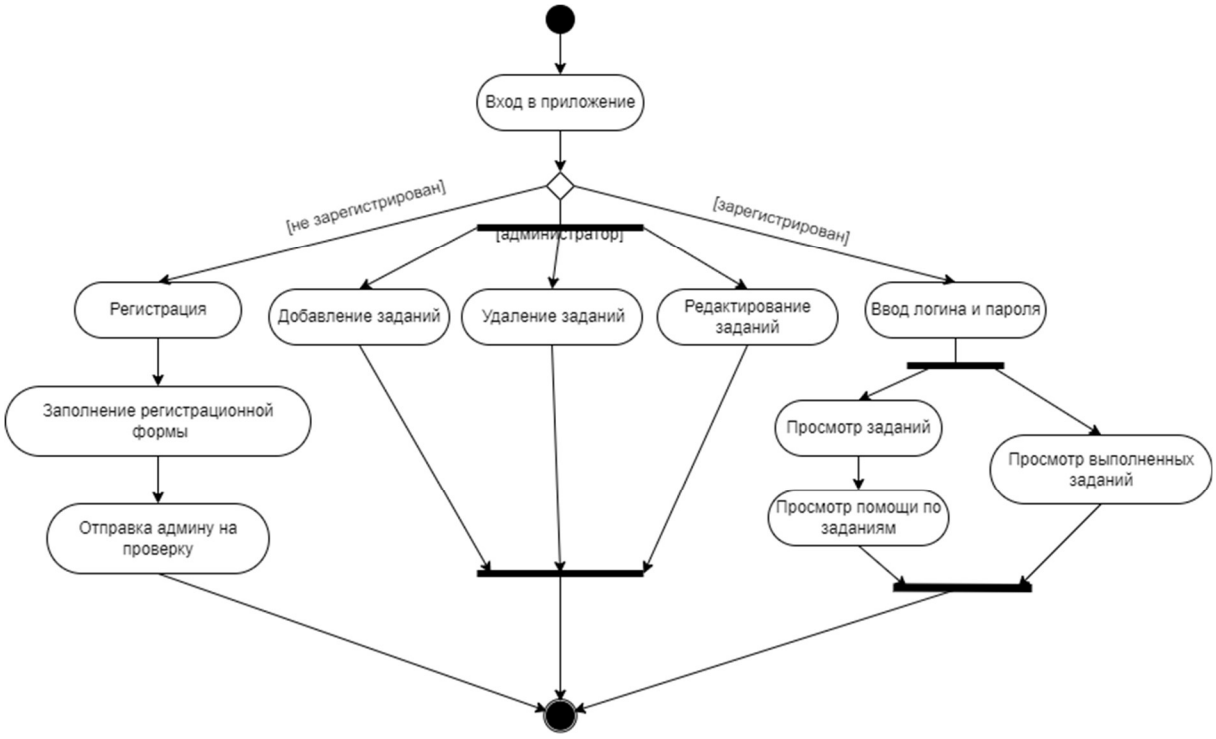

Рисунок 4 – Графическое изображение диаграммы деятельности

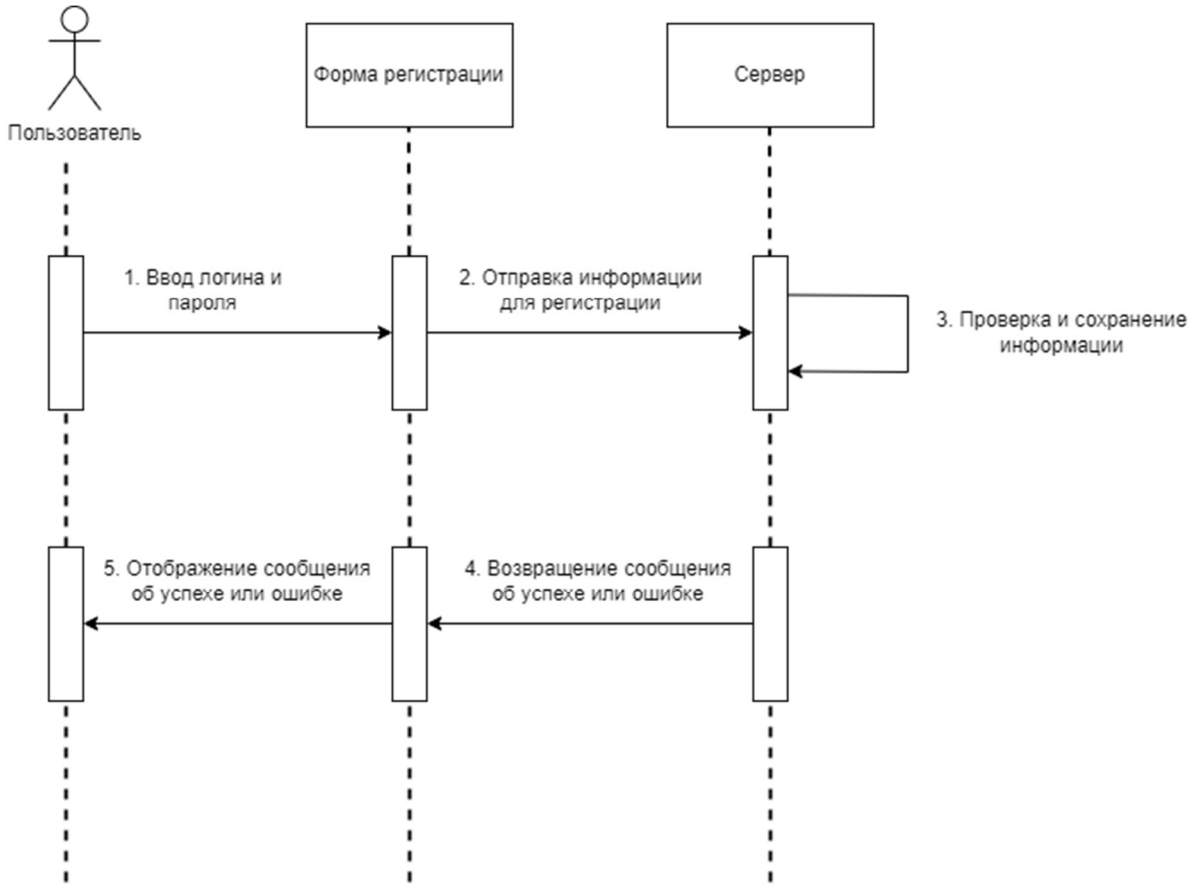

Рисунок 5 – Графическое изображение диаграммы последовательности

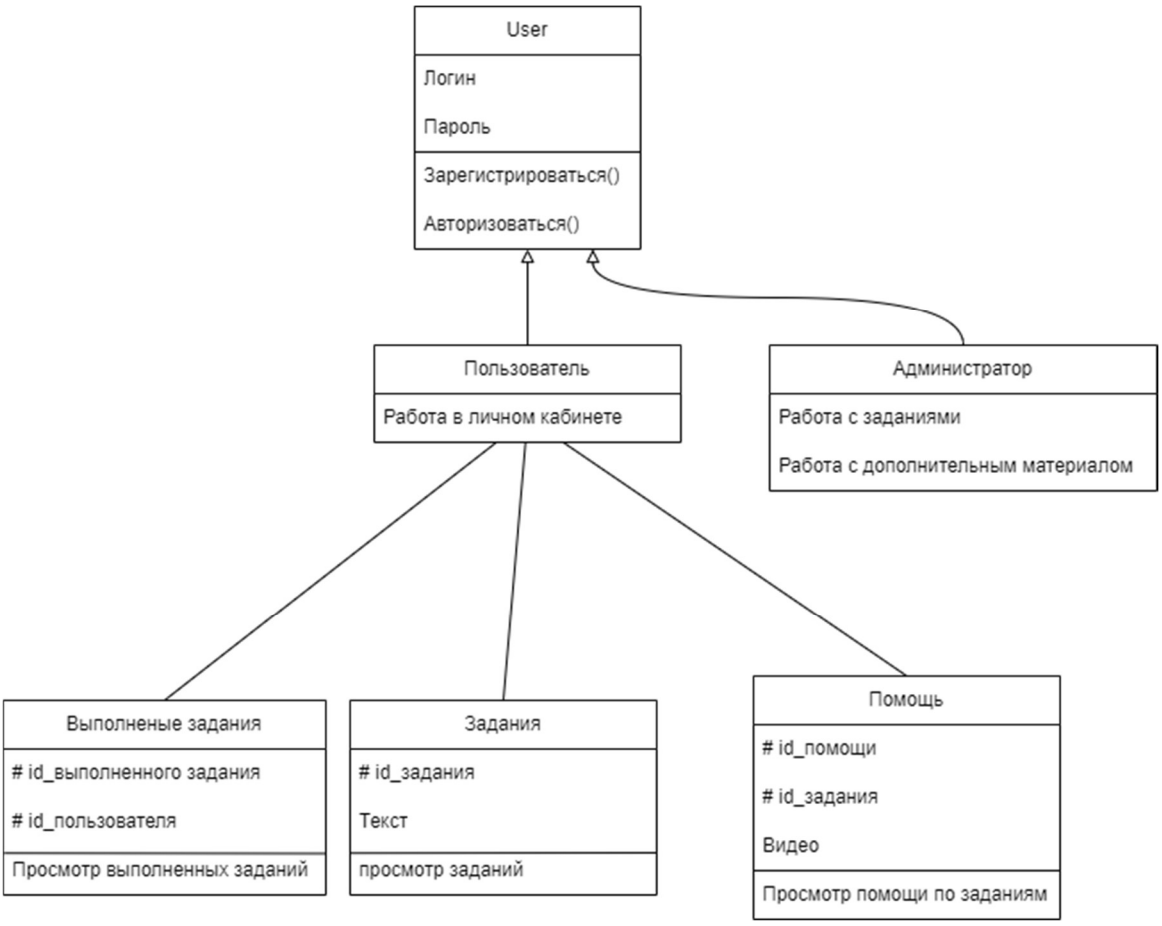

Рисунок 6 – Графическое изображение диаграммы классов

Рисунок 7 – Графическое изображение диаграммы объектов

2.4 Проектирование пользовательского интерфейса

Поставленной задачей была реализация ux/ui интерфейсов. При разработке интерфейсов были использованы следующие разрешения: 1440px, 768px, 320px. При создании UX/UI интерфейсов были использованы модульные сетки с целью создания пропорционального, понятного интерфейса. Использовались преимущественно оттенки белого и фиолетового цветов. Основные разделы доступны с первой страницы. Таким образом был реализован понятный пользовательский интерфейс.

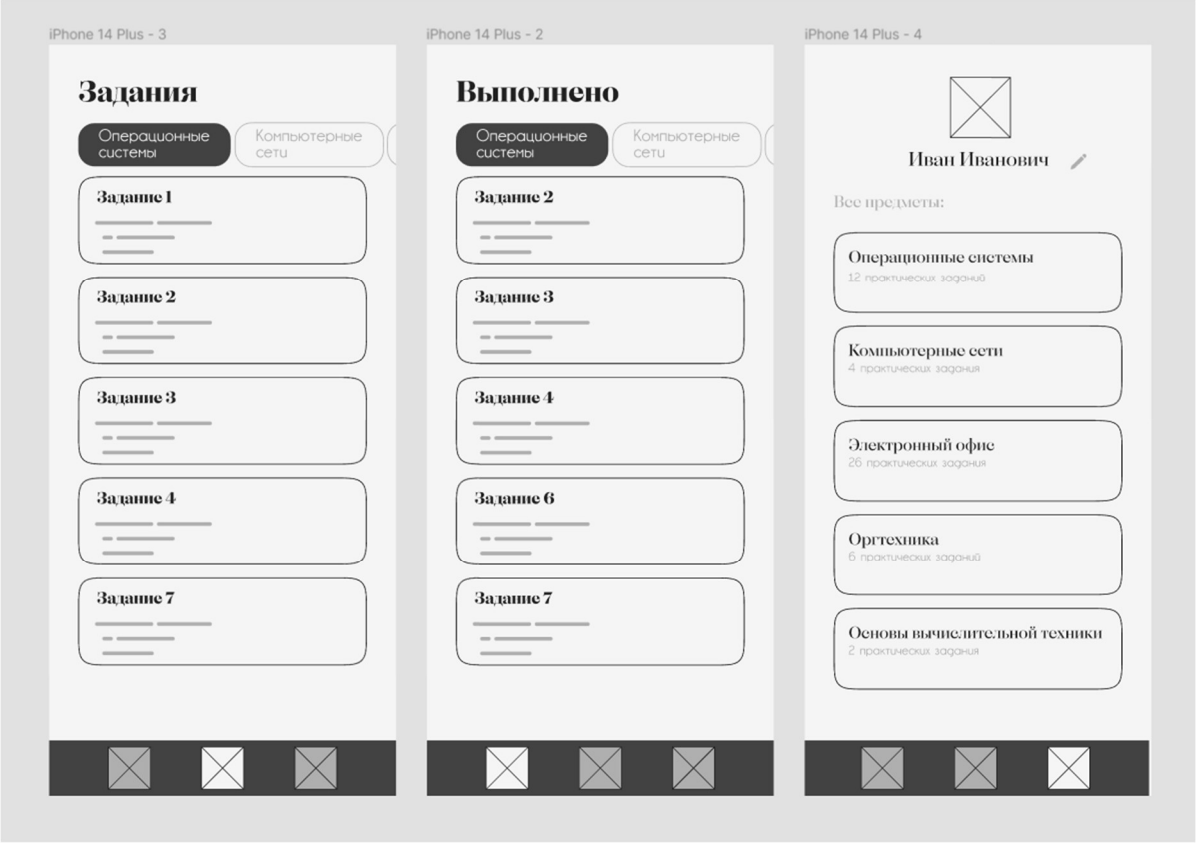

Рисунок 8 – UX интерфейс программы

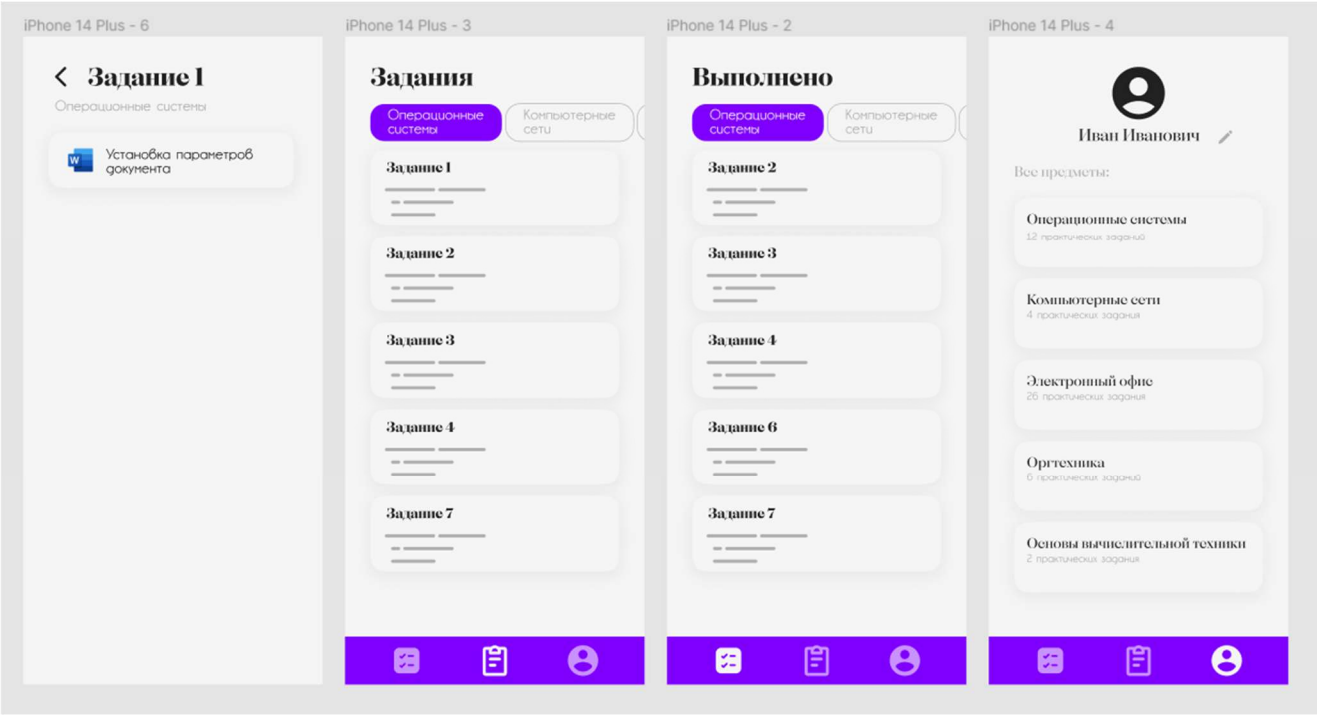

Рисунок 7 – UI интерфейс программы

3 Реализация

3.1 Руководство программиста

3.1.1 Организация данных

В программе используется используется основной файл программы main.dart, который запускает приложение, инициализирует базу данных и запускает файл home_page.dart, в котором приложение определяет, какую страницу открыть, в зависимости от того, зарегистрирован ли пользователь. В приложении подключена библиотека Hive, которая позволяет сохранять все данные, вписанные пользователем, на устройстве.

Файл register_page открывается, если пользователь не зарегистрирован в приложении. На этом экране реализована регистрация через поля ввода: ввод логина, пароля и повторный ввод пароля.

Файл task_page.dart является первым экраном, после прохождения регистрации пользователем. В нём расположены кнопки для выбора предметов, а также задания, для каждого предмета, на которые также можно нажать и перейти на страницу задания, за которую отвечает файл task_detail_page.dart.

В task_detail_page реализовано отображение выбранного задания и его информация. Также здесь реализована кнопка отвечающая за функцию отметки задания, как выполнено, или не выполнено.

Файл doneTask_page.dart отображает все выполненные задания определённого предмета, который можно также выбрать, как и на экране заданий.

Файл subjects_list.dart является классом, в котором описывается список с элементами: name, pressed, tasks.

Файл task_list.dart также является классом, в котором описан список с элементами: name, description, difficulty, finished.

Файл user_list.dart описывает класс, в котором есть список с элементами: name, password.

Файл data_base.dart подключает библиотеку Hive к спискам, с элементами, описанными выше.

3.1.2 Структура программы

На главной странице присутствует нижнее меню, содержащее в себе кнопки для перехода на другие страницы приложения.

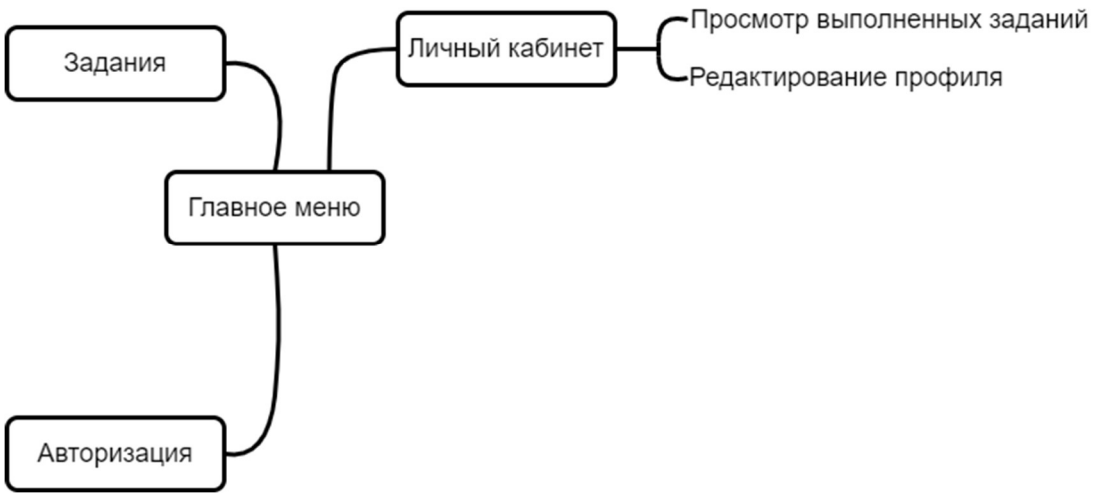

Рисунок 8 – графическое изображение системы меню приложения

4 Тестирование

4.1 Тесты на использование

В ходе разработки программного продукта были составлены тесты, которые необходимо выполнить в дальнейшем. Тесты составлены таким образом, чтобы предусмотреть максимальное количество возможных действий.

Таблица 6 – проведение тестов

№ теста	Тест	Действия	Ожидаемый результат	Физический результат
1	Проверка страницы Регистрации	1. Запустить приложение 2. Проверить наличие страницы Регистрации	Отображается страница Регистрации с полями для ввода логина, пароля и подтверждения пароля	Отображается страница Регистрации с необходимыми полями
2	Регистрация с корректными данными	1. Ввести логин, пароль и подтверждение пароля 2. Нажать кнопку регистрации	Пользователь успешно перенаправлен на страницу Заданий	Пользователь перешел на страницу Заданий
3	Регистрация с неподтвержденным паролем	1. Ввести логин и пароль 2. Не подтверждать пароль 3. Нажать кнопку регистрации	Появляется сообщение об ошибке о несовпадении паролей	Отображается сообщение об ошибке
4	Переход на страницу задания	1. Выбрать предмет на странице Заданий 2. Нажать на задание	Пользователь перенаправлен на страницу задания с возможностью отметить его выполнение	Пользователь перешел на страницу задания
5	Отметка задания как выполненного	1. Отметить задание как выполненное	Задание появляется на странице выполненных заданий	Задание появилось на странице выполненных заданий
6	Попытка перехода на выполненное задание	1. Попытаться открыть выполненное задание	Задание не доступно для отметки или просмотра	Задание не доступно для отметки или просмотра

Продолжение таблицы 6

7	Просмотр списка предметов и количества заданий на странице Аккаунт	1. Перейти на страницу Аккаунт	Отображается список всех предметов и количество заданий в каждом предмете	Отображается список предметов и их количество заданий
8	Смена логина пользователя	1. Нажать кнопку смены логина 2. Ввести новый логин 3. Нажать кнопку Ок	Новый логин отображается на странице Аккаунт	Новый логин отображается на странице Аккаунт
9	Отмена смены логина	1. Нажать кнопку смены логина 2. Не вводить новый логин 3. Нажать кнопку Отмена	Всплывающее окно закрывается без изменений логина	Всплывающее окно закрылось без изменений

4.2 Отчёт о результатах тестирования

Таблица 7 – результаты тестов

№	Статус
1	Выполнено успешно
2	Выполнено успешно
3	Выполнено успешно
4	Выполнено успешно
5	Выполнено успешно
6	Выполнено успешно
7	Выполнено успешно
8	Выполнено успешно
9	Выполнено успешно

5 Руководство пользователя

5.1 Общие сведения

Наименованием приложения является «StudyPal».

Назначение – программный продукт разрабатывался для учеников, которым нужно просмотреть задание, выполнить его, а также получить помощь по выполнению задания. В приложении пользователь может посмотреть список предметов и задания, вложенные в предметы. Также пользователь может просмотреть помощь по заданию.

5.2 Руководство

Для инсталяции приложения нужно запустить файл APK после чего появится окно установки, в котором нужно нажать на кнопку «Установить»

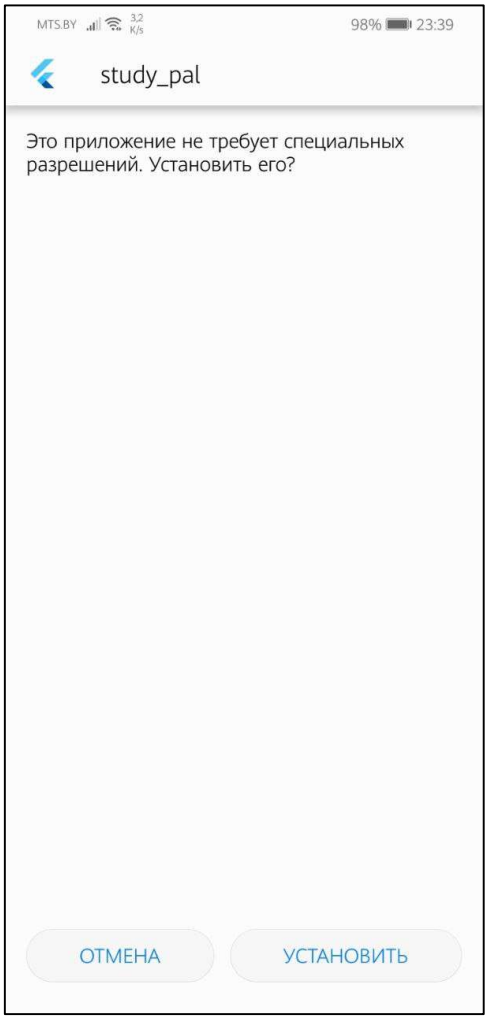

Рисунок 9 – окно инсталяции приложения

5.3 Запуск приложения

Для того, чтобы запустить данное приложение нужно нажать на его иконку, на экране смартфона.

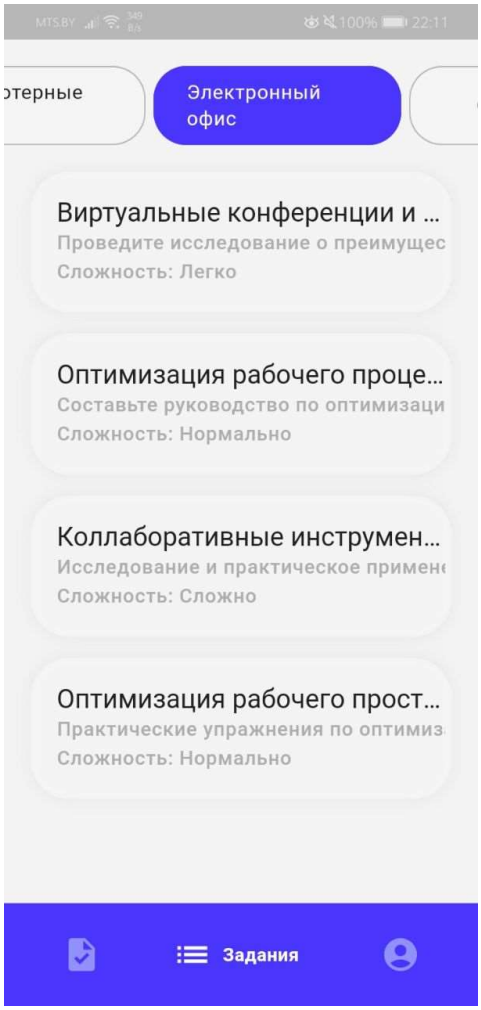

Рисунок 10 – главный экран приложения

5.4 Инструкция по работе с конфигурацией

Данное приложение поддерживает такие функции, как: регистрация, отметка заданий как выполненные, смена логина пользователя.

При первом включении приложения появляется окно регистрации, на котором пользователь может зарегистрироваться.

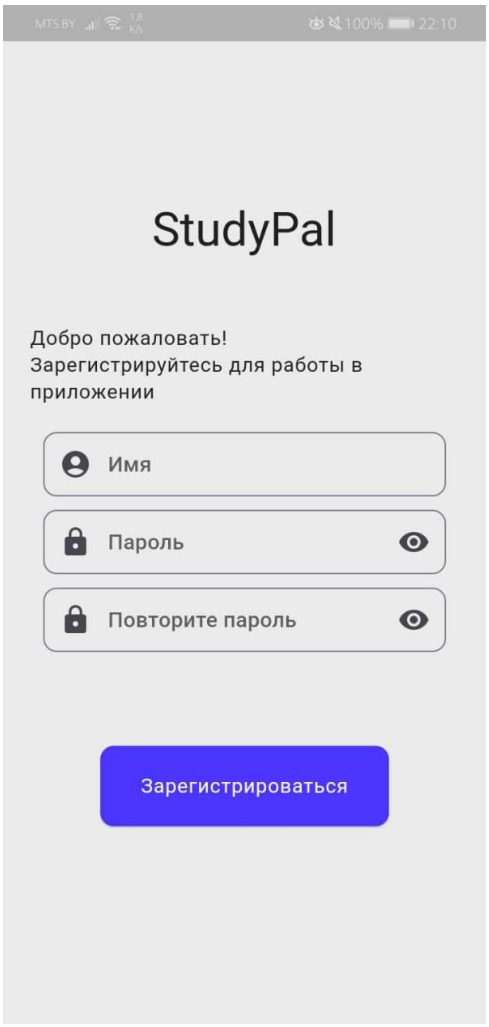

Рисунок 11 – экран регистрации

При нажатии на иконку глаза можно включить, или отключить скрытие пароля.

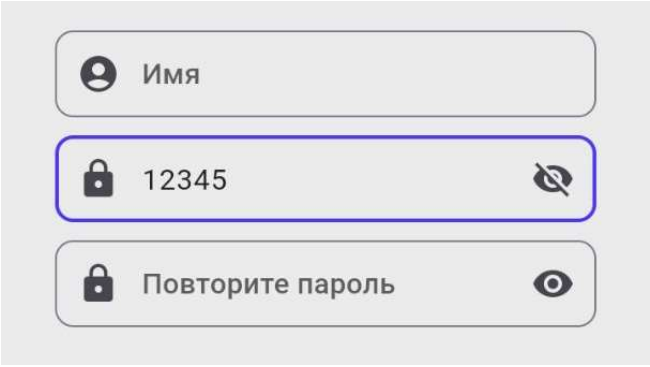

Рисунок 12 – поля ввода на экране регистрации

После регистрации пользователь попадает на главный экран, где может выбирать предметы в панели предметов, сверху.

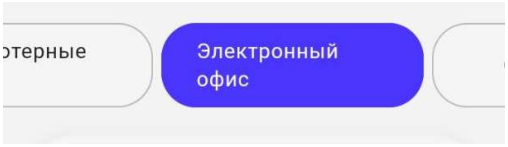

Рисунок 13 – панель предметов

При нажатии на одно из заданий на главном экране пользователь попадает на экран этого задания.

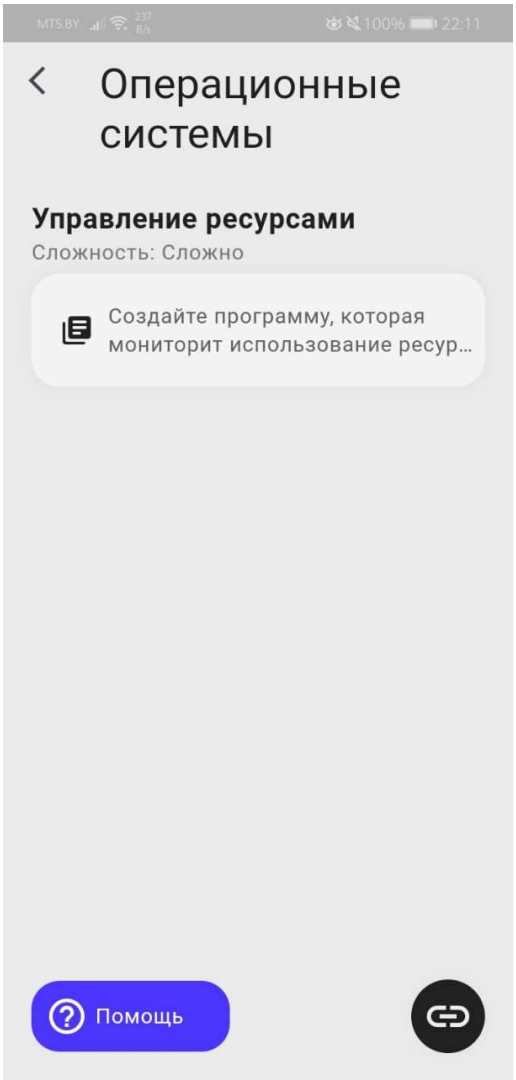

Рисунок 14 – экран задания

При нажатии на кнопку ссылки пользователь может пометить задание как выполненное. При этом кнопка сменит свой вид на белую галочку, на зелёном фоне.

Рисунок 15 – кнопка ссылки

Рисунок 16 – кнопка выполненного задания

При нажатии на кнопку «назад» приложение перейдёт обратно на главный экран.

По нажатии на кнопку «Выполненные задания» нижнего меню пользователь переходит на экран выполненных заданий, где также, как и наглавном экране может переключаться между предметами, при помощи панели предметов.

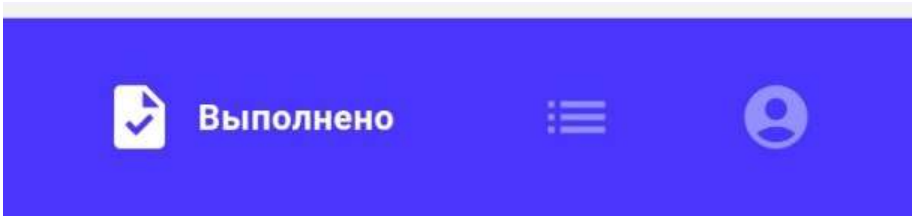

Рисунок 17 – нижнее меню приложения

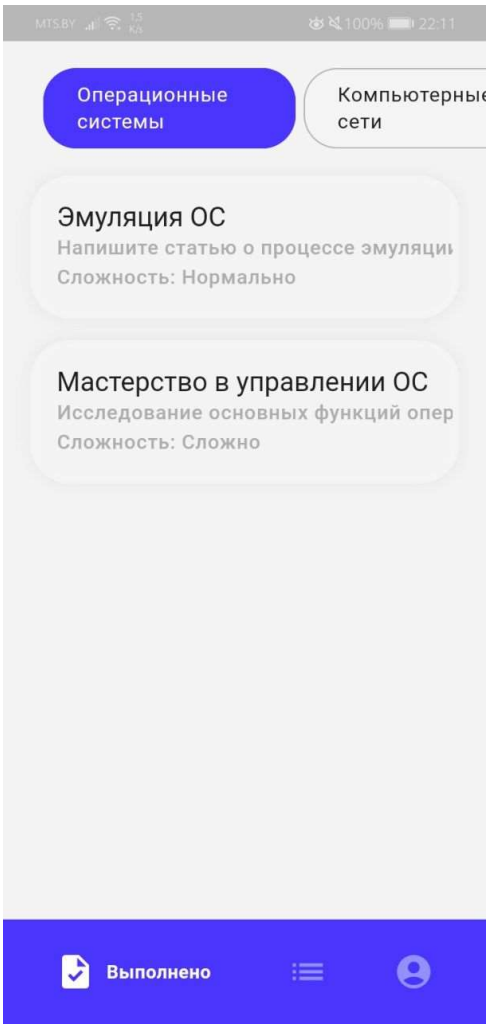

Рисунок 18 – экран «выполненные задания»

По нажатии на кнопку «Аккаунт» в нижнем меню приложения, пользователь переходит на экран аккаунта, где может посмотреть все свои предметы, и количество заданий в каждом из предметов.

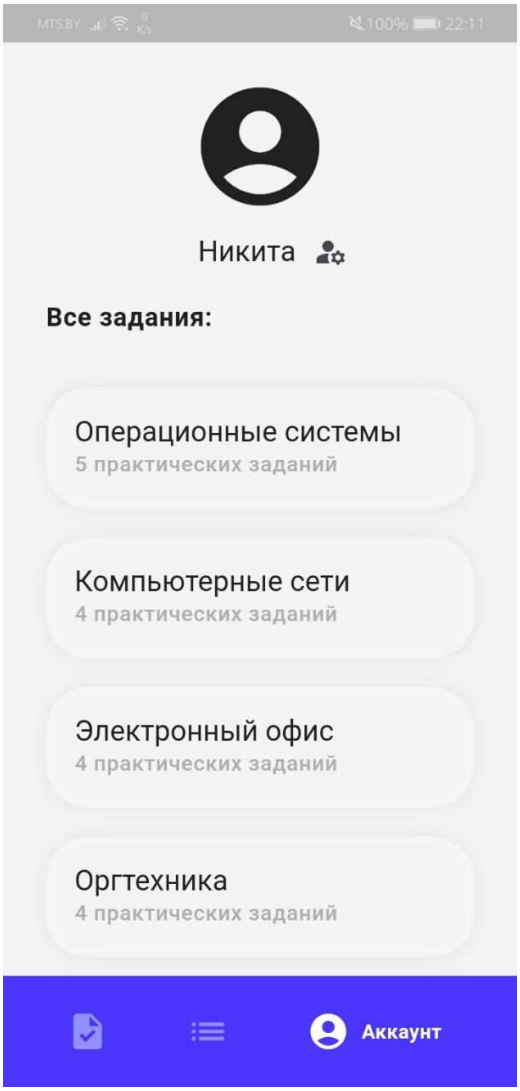

Рисунок 19 – экран аккаунта

При нажатии на кнопку настройки аккаунта пользователь может сменить логин в сплывающем окне смены логина.

Рисунок 20 – кнопка настройки аккаунта

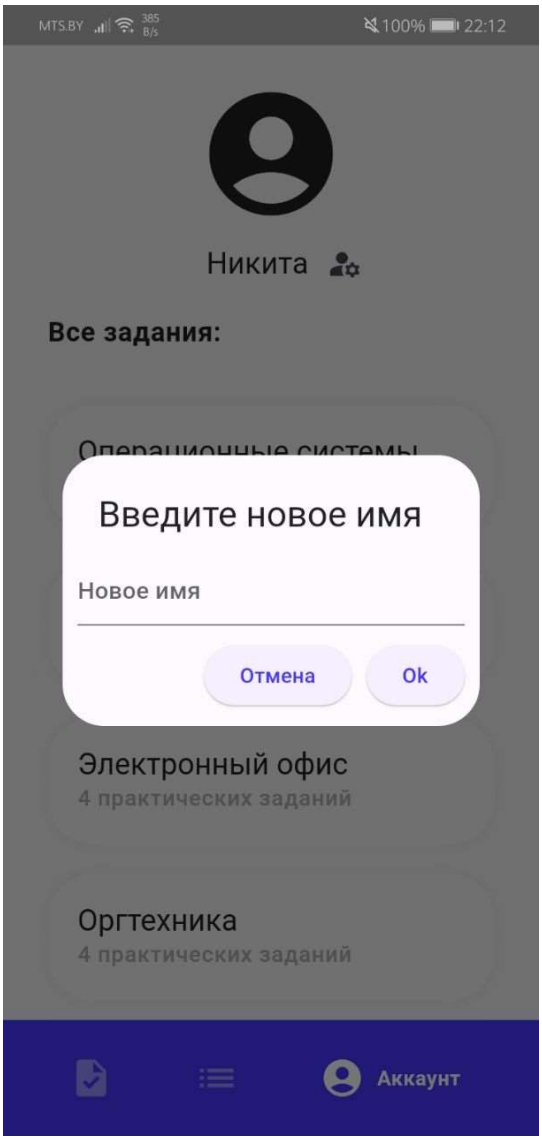

Рисунок 21 – всплывающее окно смены логина

5.5 Завершение работы с приложением

Завершить работу с приложением можно с помощью кнопок навигации системы мобильного устройства.

Заключение

Целью данного учебного проекта являлась разработка мобильного приложения «StudyPal».

В ходе реализации поставленной задачи был изучен язык Dart и фреймворк Flutter, были укреплены знания графических редакторов.

Следует также указать, что в поставленной задаче был реализован простой интефейс, который позволяет использовать приложение пользователю, не обладающему дополнительными знаниями ЭВМ. Также основной функционал реализован для пользователя, функционал для администратора не выполняется.

После тщательного тестирования приложения были выявлены некоторые недоработки, которые были полностью исправлены на стадии проектирования или полностью исключены на стадии тестирования программы. В целлом, при реализации приложения, были выполнены все условия, перечисленные в предыдущих разделах пояснительной записки. Таким образом, можно сказать, что приложение было реализовано успешно.

Список используемых источников

1 Сайт Metanit.com – Режим доступа: https://metanit.com/dart/flutter/1.1.php – Дата доступа: 07.01.2024.

2 Сайт stackoverflow.com – Режим доступа: https://stackoverflow.com – Дата доступа 29.12.2023.

Листинг программы

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:hive_flutter/hive_flutter.dart';
import 'package:study_pal/Pages/register_page.dart';
import 'package:study_pal/home_page.dart';
import 'package:study_pal/models/subjects_list.dart';
import 'package:study_pal/models/task_list.dart';
import 'package:study_pal/models/user_list.dart';

void main() async {
  await Hive.initFlutter();
  Hive.registerAdapter(SubjectAdapter());
  Hive.registerAdapter(TasksAdapter());
  Hive.registerAdapter(UserAdapter());
  var box = await Hive.openBox('myBox');

  var isUserRegistered = box.containsKey('userList');

  runApp(StudyPal(isUserRegistered: isUserRegistered));
}


class StudyPal extends StatelessWidget {
  final bool isUserRegistered;

  const StudyPal({Key? key, required this.isUserRegistered}) : super(key: key);


  @override
  Widget build(BuildContext context) {
    return MaterialApp(

      routes: {
      '/HomePage': (context) => const HomePage(),
      '/RegisterPage':(context) => const RegisterPage()
    },

      debugShowCheckedModeBanner: false,

      theme: ThemeData(
        useMaterial3: true,

        colorScheme: ColorScheme.fromSeed(seedColor: const
Color.fromARGB(255, 55, 0, 255)),
```

```
        scaffoldBackgroundColor: const Color.fromARGB(245, 245, 245, 245),
        disabledColor: const Color.fromARGB(255, 146, 146, 255),
        // primaryColor: const Color.fromARGB(255, 58, 0, 232),
        shadowColor: const Color(0x66DBDBDB),
        canvasColor: const Color.fromARGB(245, 245, 245, 245),

        textTheme: GoogleFonts.latoTextTheme().copyWith(
          displayMedium: GoogleFonts.yesevaOne(
            color: const Color(0xFF202020),
            fontSize: 20,
          ),

          labelMedium: GoogleFonts.montserrat(
            color: const Color(0xFF202020),
            fontSize: 15,
            fontWeight: FontWeight.normal
          ),

          labelSmall: GoogleFonts.montserrat(
            color: const Color(0xFFAEAEAE),
            fontSize: 15,
          )
        )
      ),

      home: isUserRegistered ? const HomePage() : const RegisterPage(),
    );
  }
}
```

```
import 'package:flutter/material.dart';
import 'package:google_nav_bar/google_nav_bar.dart';
import 'Pages/account_page.dart';
import 'Pages/doneTask_page.dart';
import 'Pages/task_page.dart';

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {

  final List<Widget> pages = [
    const DoneTask(),
    const TaskPage(),
    const AccountPage()
  ];

  int selectedIndex = 1;

  void navigationBarIndex(int index) {
    setState(() {
      selectedIndex = index;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(

      bottomNavigationBar: Container(
        padding: const EdgeInsets.symmetric(vertical: 14),

        decoration: BoxDecoration(
          color: Theme.of(context).colorScheme.primary,

        ),

        child: GNav(

          backgroundColor: Theme.of(context).colorScheme.primary,
```

```
            color: Theme.of(context).disabledColor,
            activeColor: Colors.white,
            iconSize: 30,
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            gap: 8,
            selectedIndex: selectedIndex,
            padding: const EdgeInsets.all(10),
            // tabBackgroundColor:const Color.fromARGB(255, 55, 0, 253),

            onTabChange: (index) {
              navigationBarIndex(index);
            },

            tabs: const [
              GButton(icon: Icons.task, text: 'Выполнено'),

              GButton(icon: Icons.list, text: 'Задания'),

              GButton(icon: Icons.account_circle_rounded, text: 'Аккаунт'),
            ],
          ),
        ),

        body: pages[selectedIndex],

      );
    }
  }
```

```
import 'package:flutter/material.dart';
import 'package:hive_flutter/hive_flutter.dart';
import 'package:study_pal/Pages/task_detail_page.dart';
import '../Data/data_base.dart';

class TaskPage extends StatefulWidget {
  const TaskPage({Key? key}) : super(key: key);

  @override
  State<TaskPage> createState() => _TaskPageState();
}

class _TaskPageState extends State<TaskPage> {
  final _myBox = Hive.box('myBox');
  late toDoDataBase db = toDoDataBase();

  @override
  void initState() {
    super.initState();

    if (_myBox.get('subjectList') == null || _myBox.get('userList') == null) {
      db.createInitialData();
    } else {
      db.loadData();
    }
  }

  void pressedButton(int index) {
    setState(() {
      for (int i = 0; i < db.subjects.length; i++) {
        db.subjects[i].pressed = i == index;
      }
    });

  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: ListView(
        children: <Widget>[
          Column(
            children: <Widget>[
              ConstrainedBox(
```

```
constraints: const BoxConstraints(maxHeight: 100),

child: ListView.builder(
  padding: const EdgeInsets.only(bottom: 20, top: 20, left: 30),
  shrinkWrap: true,
  physics: const ScrollPhysics(),
  scrollDirection: Axis.horizontal,
  itemCount: db.subjects.length,
  itemBuilder: (BuildContext context, index) {

    return Container(
      margin: const EdgeInsets.only(right: 6),
      width: 188,
      child: OutlinedButton(
        onPressed: () {
          pressedButton(index);
        },
        style: OutlinedButton.styleFrom(
          backgroundColor: db.subjects[index].pressed
              ? Theme.of(context).colorScheme.primary
              : const Color.fromARGB(245, 245, 245, 245),
          side: BorderSide(
              color: db.subjects[index].pressed
                  ? Theme.of(context).colorScheme.primary
                  : Colors.black26),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(25),
          ),
          padding: const EdgeInsets.symmetric(horizontal: 25),
        ),
        child: Text(
          db.subjects[index].name,
          textAlign: TextAlign.left,
          overflow: TextOverflow.ellipsis,
          softWrap: true,
          maxLines: 2,
          style: Theme.of(context).textTheme.labelMedium!.copyWith(
                color: db.subjects[index].pressed
                    ? Colors.white
                    : Colors.grey[900],
          ),
        ),
      ),
    );
```

```
                  },
                ),
              ),
              ListView.builder(
                shrinkWrap: true,
                physics: const NeverScrollableScrollPhysics(),
                itemCount: db.subjects.length,
                itemBuilder: (BuildContext context, subjectIndex) {

                  final subject = db.subjects[subjectIndex];

                  return Visibility(
                    visible: subject.pressed,

                    child: ListView.builder(
                      reverse: true,
                      padding: const EdgeInsets.symmetric(horizontal: 20),
                      shrinkWrap: true,
                      physics: const NeverScrollableScrollPhysics(),
                      itemCount: subject.tasks.length,
                      itemBuilder: (BuildContext context, taskIndex) {

                        final task = subject.tasks[taskIndex];

                        return GestureDetector(
                          onTap: () {
                            Navigator.push(context,  MaterialPageRoute(builder:(context) => TaskDetailPage(task:  db.subjects[subjectIndex].tasks[taskIndex],  subject: db.subjects[subjectIndex], subjectIndex: subjectIndex, taskIndex: taskIndex,),));
                          },
                          child: Container(
                            padding:  const  EdgeInsets.only(right:  5,  left:  20,  top:  20, bottom: 20),
                            margin: const EdgeInsets.only(bottom: 16),
                            decoration: BoxDecoration(
                              color: Theme.of(context).canvasColor,
                              borderRadius: BorderRadius.circular(30),
                              boxShadow: [
                                BoxShadow(
                                  color: Theme.of(context).shadowColor,
                                  blurRadius: 10,
                                  spreadRadius: 2,
                                ),
                              ],
```

```
                ),
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                  children: [

                    Text(
                      task.name,
                      style: Theme.of(context).textTheme.displayMedium,

                      softWrap: true,
                      overflow: TextOverflow.ellipsis,
                    ),

                    Text(
                      task.description,
                      style: Theme.of(context).textTheme.labelSmall,
                      softWrap: false,
                    ),
                    Text(
                      "Сложность: ${task.difficulty}",
                      style: Theme.of(context).textTheme.labelSmall,
                    ),
                  ],
                ),
              ),
            );
          },
        ),
      );
    },
  ),
],
),
),
),
),
);
}
}
```

```
import 'package:flutter/material.dart';
import 'package:hive_flutter/adapters.dart';
import 'package:study_pal/models/subjects_list.dart';
import 'package:study_pal/models/task_list.dart';

import '../Data/data_base.dart';

class TaskDetailPage extends StatefulWidget {

  final Tasks task;
  final Subject subject;
  final int taskIndex;
  final int subjectIndex;

  const TaskDetailPage({Key? key, required this.task, required this.subject,
required this.taskIndex, required this.subjectIndex}) : super(key: key);

  @override
  State<TaskDetailPage> createState() => _TaskDetailPageState();
}

class _TaskDetailPageState extends State<TaskDetailPage> {

  final _myBox = Hive.box('myBox');
  late toDoDataBase db = toDoDataBase();
  bool isDone = true;

  @override
  void initState() {
    super.initState();

    if (_myBox.get('subjectList') == null || _myBox.get('userList') == null) {
      db.createInitialData();
    } else {
      db.loadData();
    }
  }

  void changeTask(bool done, int taskIndex, int subjectIndex) {
    setState(() {
      db.subjects[subjectIndex].tasks[taskIndex].finished = done;
    });

    db.updateDataBase();
```

```
}


@override
Widget build(BuildContext context) {
 return Scaffold(
  body: Column(
   mainAxisAlignment: MainAxisAlignment.spaceBetween,
   crossAxisAlignment: CrossAxisAlignment.start,

   children: [

    Column(
     crossAxisAlignment: CrossAxisAlignment.start,

     children: [
      Container(
       margin: const EdgeInsets.only(top: 30),

       child: Row(
        crossAxisAlignment: CrossAxisAlignment.baseline,
        textBaseline: TextBaseline.alphabetic,
        children: [

         IconButton(
          onPressed:() {
           Navigator.pop(context);
          },
          icon: const Icon(Icons.arrow_back_ios_new)
         ),

         const SizedBox(
          width: 20,
         ),

         Expanded(
          child: Text(
           widget.subject.name,
           softWrap: true,
           // overflow: TextOverflow.clip,
           maxLines: 3,

           style: Theme.of(context).textTheme.displayMedium!.copyWith(
            fontSize: 30
```

```
          ),
        ),
      ),
    ],
  ),
),

Container(
  padding: const EdgeInsets.only(left: 20, right: 20, top: 30),
  child: Text(
      widget.task.name,

      style: Theme.of(context).textTheme.labelMedium!.copyWith(
        fontSize: 20,
        fontWeight: FontWeight.bold
      ),
    ),
),

Container(
  margin: const EdgeInsets.only(left: 20),
  child: Text(
    'Сложность: ${widget.task.difficulty}',

    style: Theme.of(context).textTheme.labelMedium!.copyWith(
      color: Colors.grey[600]
    ),
  ),
),

Container(
  margin: const EdgeInsets.symmetric(horizontal: 20, vertical: 5),
  padding: const EdgeInsets.only(top: 20, bottom: 20, left: 20, right: 5),
  alignment: Alignment.centerLeft,

  decoration: BoxDecoration(
    color: Theme.of(context).canvasColor,
    borderRadius: BorderRadius.circular(20),
    boxShadow: [BoxShadow(
      color: Theme.of(context).shadowColor,
      blurRadius: 10,
      spreadRadius: 2,
    )]
  ),
```

```
            child: Row(
              textBaseline: TextBaseline.alphabetic,

              children: [
                const      Icon(Icons.my_library_books_rounded,      textDirection:
TextDirection.ltr,),

                const SizedBox(
                  width: 10,
                ),

                Expanded(
                  child: Text(
                    widget.task.description == '' ? 'Нажмите, чтобы открыть' :
widget.task.description,
                    textAlign: TextAlign.left,
                    softWrap: true,
                    maxLines: 2,
                    overflow: TextOverflow.ellipsis,

                    style: Theme.of(context).textTheme.labelMedium!.copyWith(
                      color: Colors.grey[700]
                    ),
                  ),
                ),
              ],
            ),
          ),
        ],
      ),

      Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,

        children: [
          Container(

            width: 140,

            padding: const EdgeInsets.symmetric(horizontal: 10, vertical: 10),
            margin: const EdgeInsets.symmetric(horizontal: 20, vertical: 20),

            decoration: BoxDecoration(
```

```
      color: Theme.of(context).colorScheme.primary,

      borderRadius: BorderRadius.circular(20)
    ),

    child: Row(
      mainAxisAlignment: MainAxisAlignment.start,
      children: [

        const Icon(Icons.help_outline, color: Colors.white, size: 30,),

        const SizedBox(width: 5,),

        Text(
          'Помощь',

          style: Theme.of(context).textTheme.labelMedium!.copyWith(
            color: Colors.white
          ),
        ),
      ],
    ),
  ),

  Container(
    margin: const EdgeInsets.only(right: 20),

    decoration: BoxDecoration(
      color: widget.task.finished ? Colors.green[400] : Colors.grey[900],
      borderRadius: BorderRadius.circular(100),

      boxShadow: [BoxShadow(
        color: Theme.of(context).shadowColor,
        blurRadius: 10,
        spreadRadius: 2
      )]
    ),

    child: IconButton(
      disabledColor: Theme.of(context).colorScheme.primary,
      color: Colors.grey[200],
      highlightColor: const Color.fromARGB(255, 0, 128, 255),
      iconSize: 36,
      onPressed:() {
```

```
                    if (widget.task.finished != isDone) {
                      changeTask(isDone, widget.taskIndex, widget.subjectIndex);
                    }
                    else {
                      isDone = !isDone;
                      changeTask(isDone, widget.taskIndex, widget.subjectIndex);
                    }
                  },
                  icon: widget.task.finished ? const Icon(Icons.done) : const
Icon(Icons.link)
                ),
              )
            ],
          ),
        ],
      ),
    );
  }
}
```

```
import 'package:flutter/material.dart';
import 'package:hive_flutter/adapters.dart';
import '../Data/data_base.dart';
import '../models/user_list.dart';

class RegisterPage extends StatefulWidget {
  const RegisterPage({super.key});

  @override
  State<RegisterPage> createState() => _RegisterPageState();
}

class _RegisterPageState extends State<RegisterPage> {

  final _myBox = Hive.box('myBox');
  late toDoDataBase db = toDoDataBase();
  final nameController = TextEditingController();
  final passwordController = TextEditingController();
  final passwordCheckController = TextEditingController();
  bool passwordTextObscure = true;
  bool passwordCheckObscure = true;
  double radius = 10;

  @override
  void initState() {
    super.initState();

    if (_myBox.get('subjectList') == null || _myBox.get('userList') == null) {
      db.createInitialData();
    } else {
      db.loadData();
    }
  }

  void addAccount(String name, String password) {
    setState(() {
      // db.users[0].name = name;
      // db.users[0].password = password;
      db.users.add(User(name: name, password: password));
      // db.users.clear();
    });

    db.updateDataBase();
  }
```

```
@override
Widget build(BuildContext context) {
 return Scaffold(
  resizeToAvoidBottomInset: false,
  body: Column(
   mainAxisAlignment: MainAxisAlignment.center,
   children: [

    Center(
     child: Text(
      'StudyPal',

      style: Theme.of(context).textTheme.displayMedium!.copyWith(
       fontSize: 35
      ),
     ),
    ),

    const SizedBox(
     height: 50,
    ),

    Container(

     padding: const EdgeInsets.only(left: 20, right: 5),

     child: Text(
      'Добро пожаловать!\nЗарегистрируйтесь для работы в приложении',

      style: Theme.of(context).textTheme.labelMedium,
     ),
    ),


    Container(
     margin: const EdgeInsets.only(top: 20),
     padding: const EdgeInsets.symmetric(horizontal: 30),

     child: TextFormField(
      controller: nameController,
      textCapitalization: TextCapitalization.sentences,
```

```
            decoration: InputDecoration(
              contentPadding: const EdgeInsets.symmetric(horizontal: 20),
              hintText: 'Имя',

              border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(radius)
              ),

              prefixIcon: const Icon(Icons.account_circle_rounded)
            ),

          ),
        ),

        Container(
          margin: const EdgeInsets.only(top: 10),
          padding: const EdgeInsets.symmetric(horizontal: 30),

          child: TextFormField(
            controller: passwordController,
            obscureText: passwordTextObscure,

            decoration: InputDecoration(
              contentPadding: const EdgeInsets.symmetric(horizontal: 20),
              hintText: 'Пароль',

              border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(radius)
              ),

              suffixIcon: IconButton(
                onPressed:() {
                  setState(() {
                    passwordTextObscure = !passwordTextObscure;
                  });
                },
                icon: Icon(passwordTextObscure ? Icons.visibility :
Icons.visibility_off)
              ),

              prefixIcon: const Icon(Icons.lock)
            ),
          ),
```

```
        ),

        Container(
          margin: const EdgeInsets.only(top: 10),
          padding: const EdgeInsets.symmetric(horizontal: 30),

          child: TextFormField(
            controller: passwordCheckController,
            obscureText: passwordCheckObscure,

            decoration: InputDecoration(
              contentPadding: const EdgeInsets.symmetric(horizontal: 20),
              hintText: 'Повторите пароль',

              border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(radius)
              ),

              suffixIcon: IconButton(
                onPressed:() {
                  setState(() {
                    passwordCheckObscure = !passwordCheckObscure;
                  });
                },
                icon: Icon(passwordCheckObscure ? Icons.visibility : Icons.visibility_off)
              ),

              prefixIcon: const Icon(Icons.lock)
            ),
          ),
        ),

        const SizedBox(
          height: 70,
        ),

        ElevatedButton(
          onPressed:() {
            if(passwordController.text == passwordCheckController.text && nameController.text != ''  && passwordController.text != '') {
              addAccount(nameController.text, passwordController.text);
              Navigator.pushNamed(context, '/HomePage');
            }
```

```
            },

            style: ButtonStyle(
              backgroundColor:
MaterialStatePropertyAll(Theme.of(context).colorScheme.primary),
              overlayColor:
MaterialStatePropertyAll(Theme.of(context).colorScheme.primary),
              padding:                                                    const
MaterialStatePropertyAll(EdgeInsets.symmetric(vertical: 20, horizontal: 30)),
              shape:
MaterialStatePropertyAll(RoundedRectangleBorder(borderRadius:
BorderRadius.circular(radius))),
              ),

            child: Text(
              'Зарегистрироваться',

              style: Theme.of(context).textTheme.labelMedium!.copyWith(
                color: Colors.white
              )
            ),
          ),
        ],
      ),
    );
  }
}
```

```
// ignore_for_file: file_names

import 'package:flutter/material.dart';
import 'package:hive_flutter/hive_flutter.dart';
import 'package:study_pal/models/task_list.dart';
import '../Data/data_base.dart';

class DoneTask extends StatefulWidget {
  const DoneTask({Key? key}) : super(key: key);

  @override
  State<DoneTask> createState() => _DoneTaskState();
}

class _DoneTaskState extends State<DoneTask> {
  final _myBox = Hive.box('myBox');
  late toDoDataBase db = toDoDataBase();

  @override
  void initState() {
    super.initState();

    if (_myBox.get('subjectList') == null || _myBox.get('userList') == null) {
      db.createInitialData();
    } else {
      db.loadData();
    }
  }

  void pressedButton(int index) {
    setState(() {
      for (int i = 0; i < db.subjects.length; i++) {
        db.subjects[i].pressed = i == index;
      }
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: ListView(
        children: <Widget>[
          Column(
            children: <Widget>[
```

```
ConstrainedBox(
  constraints: const BoxConstraints(maxHeight: 100),
  child: ListView.builder(
    padding: const EdgeInsets.only(bottom: 20, top: 20, left: 30),
    shrinkWrap: true,
    physics: const ScrollPhysics(),
    scrollDirection: Axis.horizontal,
    itemCount: db.subjects.length,
    itemBuilder: (BuildContext context, index) {
      return Container(
        margin: const EdgeInsets.only(right: 6),
        width: 188,
        child: OutlinedButton(
          onPressed: () {
            pressedButton(index);
          },
          style: OutlinedButton.styleFrom(
            backgroundColor: db.subjects[index].pressed
                ? Theme.of(context).colorScheme.primary
                : const Color.fromARGB(245, 245, 245, 245),
            side: BorderSide(
                color: db.subjects[index].pressed
                    ? Theme.of(context).colorScheme.primary
                    : Colors.black26),
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(25),
            ),
            padding: const EdgeInsets.symmetric(horizontal: 25),
          ),
          child: Text(
            db.subjects[index].name,
            textAlign: TextAlign.left,
            overflow: TextOverflow.clip,
            maxLines: 2,
            style: Theme.of(context)
                .textTheme
                .labelMedium!
                .copyWith(
                  color: db.subjects[index].pressed
                      ? Colors.white
                      : Colors.grey[900],
                ),
          ),
        ),
```

```
                );
              },
            ),
          ),
          ListView.builder(
            padding: const EdgeInsets.symmetric(horizontal: 20),
            shrinkWrap: true,
            physics: const NeverScrollableScrollPhysics(),
            itemCount: db.subjects.length,
            itemBuilder: (BuildContext context, subjectIndex) {
              final subject = db.subjects[subjectIndex];
              final List<Tasks> finishedTasks = subject.tasks.where((task) =>
task.finished).toList();

              return Visibility(
                visible: subject.pressed,
                child: ListView.builder(
                  reverse: true,
                  shrinkWrap: true,
                  physics: const NeverScrollableScrollPhysics(),
                  itemCount: finishedTasks.length,
                  itemBuilder: (BuildContext context, taskIndex) {
                    final task = finishedTasks[taskIndex];

                    return Container(
                      padding: const EdgeInsets.only(right: 5, left: 20, top: 20, bottom:
20),
                      margin: const EdgeInsets.only(bottom: 16),
                      decoration: BoxDecoration(
                        color: Theme.of(context).canvasColor,
                        borderRadius: BorderRadius.circular(30),
                        boxShadow: [
                          BoxShadow(
                            color: Theme.of(context).shadowColor,
                            blurRadius: 10,
                            spreadRadius: 2
                          ),
                        ],
                      ),
                      child: Column(
                        crossAxisAlignment: CrossAxisAlignment.start,
                        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                        children: [
                          Text(
```

```
                  task.name,
                  style: Theme.of(context).textTheme.displayMedium,
                ),
                Text(
                  task.description,
                  style: Theme.of(context).textTheme.labelSmall,
                  softWrap: false,
                ),
                Text(
                  "Сложность: ${task.difficulty}",
                  style: Theme.of(context).textTheme.labelSmall,
                ),
              ],
            ),
          );
        },
      ),
    );
  },
),
],
),
],
),
);
}
}
```

```
import 'package:flutter/material.dart';
import 'package:hive_flutter/hive_flutter.dart';
import '../Data/data_base.dart';

class AccountPage extends StatefulWidget {
  const AccountPage({Key? key}) : super(key: key);

  @override
  State<AccountPage> createState() => _AccountPageState();
}

class _AccountPageState extends State<AccountPage> {

  final _myBox = Hive.box('myBox');
  late toDoDataBase db = toDoDataBase();

  final nameController = TextEditingController();

  @override
  void initState() {
    super.initState();

    if (_myBox.get('subjectList') == null || _myBox.get('userList') == null) {
      db.createInitialData();
    } else {
      db.loadData();
    }
  }

  void changeName(String newName) {
    setState(() {
      db.users[0].name = newName;
    });

    db.updateDataBase();
  }

  void changeNameDialog() {

    showDialog(
      context: context,
      builder:(BuildContext context) {
        return StatefulBuilder(
          builder:(context, setState) {
```

```
 return AlertDialog(
title: const Text('Введите новое имя'),
surfaceTintColor: Theme.of(context).canvasColor,

contentPadding: const EdgeInsets.all(10),

content: Column(
 mainAxisSize: MainAxisSize.min,
 mainAxisAlignment: MainAxisAlignment.spaceEvenly,

 children: [

  TextFormField(
   controller: nameController,

   decoration: const InputDecoration(
    hintText: 'Новое имя',
   ),

   validator: (value) {
    if (value == null || value.isEmpty) {
     return 'Введите новое имя';
    }

    return null;
   },
  ),

  Container(
   height: 10,
  ),

  Row(
   mainAxisAlignment: MainAxisAlignment.end,
   children: [

    ElevatedButton(
     onPressed:() {
      setState(() {
       Navigator.pop(context);
       nameController.clear();
      });
     },
```

```
                child: const Text(
                  'Отмена'
                ),
              ),

              Container(
                width: 10,
              ),

              ElevatedButton(
                onPressed:() {

                  if(nameController.text.isEmpty || nameController.text == '') {
                    return;
                  }

                  else {
                    changeName(nameController.text);
                    Navigator.pop(context);
                    nameController.clear();
                  }
                },

                child: const Text(
                  'Ok'
                ),
              ),
            ],
          ),
        ],
      ),
    );
      },
    );
  },
 );
}

@override
Widget build(BuildContext context) {
 return Scaffold(
   body: Center(
     child: ListView(
       padding: const EdgeInsets.only(top: 50, left: 30, right: 30),
```

```
children: [
 const Icon(
  Icons.account_circle_rounded,
  size: 100,
 ),
 Padding(
  padding: const EdgeInsets.only(right: 20, left: 50),
  child: Row(
   mainAxisAlignment: MainAxisAlignment.center,
   children: [
    Text(
     db.users[0].name,
     style: Theme.of(context).textTheme.displayMedium,
    ),
    IconButton(
     onPressed:() {
      setState(() {
       changeNameDialog();
      });
     },
     icon: const Icon(Icons.manage_accounts), // Поменять иконку
    ),
   ],
  ),
 ),
 Container(
  margin: const EdgeInsets.symmetric(vertical: 10),
  child: Text(
   "Все задания:",
   style: Theme.of(context).textTheme.labelMedium!.copyWith(
        fontWeight: FontWeight.bold,
        fontSize: 18,
       ),
  ),
 ),
 ListView.builder(
  itemCount: db.subjects.length,
  shrinkWrap: true,
  physics: const NeverScrollableScrollPhysics(),
  itemBuilder: (BuildContext context, index) {
   return Container(
    padding: const EdgeInsets.only(left: 20, right: 5, top: 20, bottom: 20),
    margin: const EdgeInsets.only(bottom: 20),
    decoration: BoxDecoration(
```

```
                color: Theme.of(context).canvasColor,
                borderRadius: BorderRadius.circular(30),
                boxShadow: [
                  BoxShadow(
                    color: Theme.of(context).shadowColor,
                    blurRadius: 10,
                    spreadRadius: 2
                  ),
                ],
              ),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    db.subjects[index].name,
                    style: Theme.of(context).textTheme.displayMedium,
                  ),
                  Text(
                    "${db.subjects[index].tasks.length} практических заданий",
                    softWrap: false,
                    style: Theme.of(context).textTheme.labelSmall,
                  )
                ],
              ),
            );
          },
        ),
      ],
    ),
  ),
);
  }
}
```