МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 “Программное обеспечение информационной

безопасности мобильных систем”

Специализация Инженер-программист

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

Приложение для планирования и ведения проектов

Выполнил студент Тихон Алексей Александрович

(Ф.И.О.)

Руководитель проекта ст. преподаватель Блинова Е.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: ст. преподаватель Блинова Е.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: ст. преподаватель Блинова Е.А (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Оглавление

[**Введение** 3](#_Toc135211169)

[**1 Обзор аналогов и постановка задачи** 4](#_Toc135211170)

[1.1 Обзор приложения «Trello» 4](#_Toc135211171)

[1.2 Постановка задач 5](#_Toc135211172)

[1.3 Выводы по главе «Обзор аналогов и постановка задачи» 5](#_Toc135211173)

[**2. Проектирование программного продукта** 6](#_Toc135211174)

[2.1 Проектирование глобальной базы данных 6](#_Toc135211175)

[2.2 Проектирование локальной базы данных 7](#_Toc135211176)

[2.3 Проектирование сервисов доступа к данным 7](#_Toc135211177)

[2.4 Проектирование мобильного приложения 8](#_Toc135211178)

[2.5 Проектирование веб приложения 12](#_Toc135211179)

[2.6 Выводы по главе «Проектирование программного продукта» 14](#_Toc135211180)

[**3 Программная реализация продукта** 15](#_Toc135211181)

[3.1 Технические средства разработки 15](#_Toc135211182)

[3.2 Разработка глобальной базы данных 16](#_Toc135211183)

[3.3 Разработка локальной базы данных 16](#_Toc135211184)

[3.4 Разработка сервера 17](#_Toc135211185)

[3.5 Разработка мобильного приложения 18](#_Toc135211186)

[3.7 Разработка веб приложения 21](#_Toc135211187)

[3.8 Выводы по главе «Программная реализация продукта» 23](#_Toc135211188)

[**4 Обеспечение безопасности информационной системы** 24](#_Toc135211189)

[4.1 Вывод по главе «Обеспечение безопасности информационной системы» 24](#_Toc135211190)

[**5. Руководство пользователя** 25](#_Toc135211191)

[5.1 Руководство пользователя мобильного приложения 25](#_Toc135211192)

[5.2 Руководство пользователя веб приложения 33](#_Toc135211193)

[**ЗАКЛЮЧЕНИЕ** 36](#_Toc135211194)

[Список использованных источников 37](#_Toc135211195)

[ПРИЛОЖЕНИЕ А 38](#_Toc135211196)

[ПРИЛОЖЕНИЕ Б 40](#_Toc135211197)

# **Введение**

Для успешного функционирования бизнеса необходимо четко следить за рабочим процессом. Рабочий процесс можно описать с помощью трех вопросов: что, как и когда необходимо сделать. В наше время задачи становятся объемнее и сложнее, удержать все в голове становится попросту невозможно. Для решения данной проблемы существуют различные системы управления рабочими процессами. Такие системы могут применяться абсолютно в любой сфере: финансы, логистика, техническое обслуживание.

Использование систем управления рабочими процессами позволяет повысить производительность и эффективность работы, а также снизить количество ошибок. Также одним из важных преимуществ использования систем управления рабочим процессом является обеспечение прозрачности выполняемых работ. Прозрачность позволяет более точно планировать и распределять ресурсы, обеспечивать согласованность внутри коллектива.

Для успешного функционирования системы управления рабочими процессами необходимо обеспечить представление задач в удобном и простом для понимания формате. Все проекты можно разделить на три категории: выполненные, выполняющиеся и ожидающие выполнения в будущем. Также необходима возможность разбивать проект на подзадачи, которые будут распределяться между участниками команды.

Разработка мобильной версии системы управления рабочими процессами связана с возможностью возникновения ситуации, в которой у сотрудника не будет доступа к компьютеру. Однако мобильные устройства всегда находятся под рукой и обеспечивают простой и быстрый доступ к приложениям.

Также необходимо предусмотреть невозможность пользователя выйти в интернет. В такой ситуации пользователь должен обладать возможностью просмотреть текущие проекты и задачи, которые он должен выполнить. То есть необходимо обеспечить автономность приложения в условиях отсутствия доступа в сеть интернет.

Подводя итог и обобщив всю вышеизложенную информацию, определим цель выполнения курсового проекта как разработку системы управления рабочим процессом, которая позволяет создавать проекты, разбивать их на подзадачи и распределять между участниками проекта. При этом необходимо разработать мобильную версию и обеспечить её автономность с помощью синхронизации внутренней базы данных с глобальной базой данных.

# **1 Обзор аналогов и постановка задачи**

## 1.1 Обзор приложения «Trello»

В качестве аналога было выбрано мобильное приложение Trello (рис 1.1).



Рисунок 1.1 – мобильное приложение Trello

Trello - облачная программа для управления проектами небольших групп, разработанная Fog Creek Software.

Trello позволяет создавать карточки, которые представляют из себя задачу, изменять статус задачу: задача может находиться в одном из трех состояний: «Сделать», «В работе», «Готово». Карточки, в зависимости от состояния, распределяются в три колонки.

Trello позволяет создавать множество проектов, называемых «досками». Каждая из досок также имеет свое состояние.

Также Trello поддерживает функцию «Чек -листы». По сути, данная функция позволяет разбивать задачи на подзадачи для более удобного оформления задачи.

Trello – одна из самых популярных на данный момент систем управления проектами.

## 1.2 Постановка задач

Делая выводы из вышеперечисленного, можно сделать заключение и сформировать задачу: разработать приложение, обладающее следующими основными функциями:

* Возможность создавать проекты;
* Возможность создавать команды для выполнения проекта;
* Возможность создания задач и подзадач;
* Возможность изменения статуса проекта.

## 1.3 Выводы по главе «Обзор аналогов и постановка задачи»

В данной главе был выполнен обзор одного из самых популярных приложений для управления проектом – Trello. На основе возможностей данного приложения были сформулированы требования к разрабатываемому приложению, а также поставлена основная задача проекта.

# **2. Проектирование программного продукта**

## 2.1 Проектирование глобальной базы данных

Глобальная база данных состоит из 7 таблиц: «AppUser», «Organisation», «OrganisationMember», «ProjectMember», «Project», «SubTask», «SubTaskExecutor».

Схема глобальной базы данных представлена на рисунке 2.1.

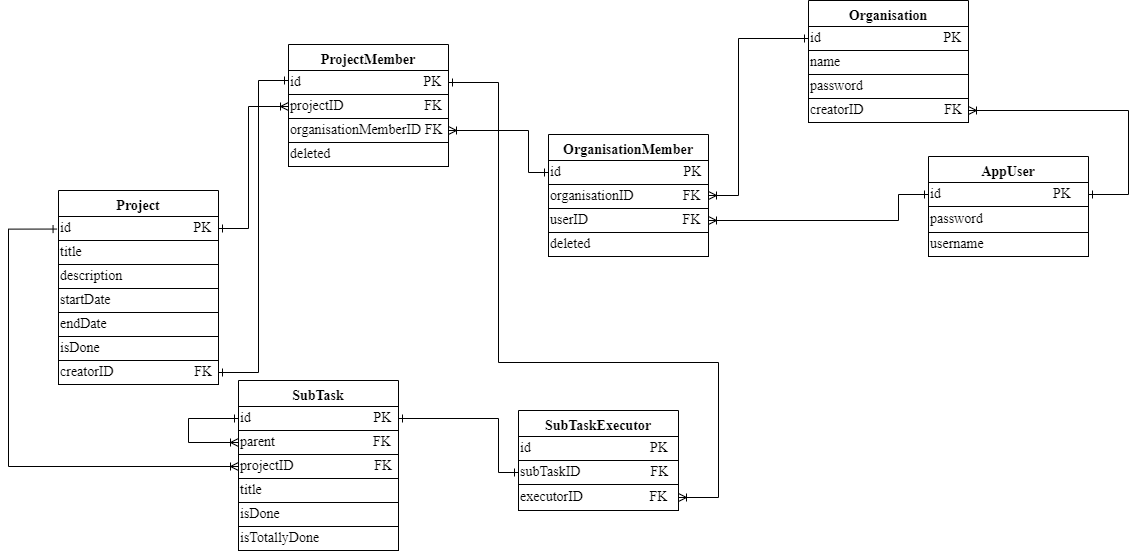


Рисунок 2.1 – схема глобальной базы данных

Таблица «AppUser» предназначена для хранения имен пользователей, а также их паролей.

Таблица «Organisation» хранит информацию об организациях, в которые могут вступать пользователи.

Таблица «OrganisationMember» используется для хранения информации об участниках организации.

Таблица «Project» хранит данные о проектах: название, описание, дата начала, дата конца, состояние проекта, а также данные о создателе проекта.

Таблица «ProjectMember» хранит информацию об участниках проекта, а также используется для идентификации исполнителя задачи.

Таблица «SubTaskExecutor» используется для хранения данных об исполнителе задачи.

Таблица «SubTask» хранит в себе данные о задачах, которые должны быть выполнены. В данной таблице столбец parent указывает на родительскую задачу.

Таблица «SubTaskExecutor» используется для хранения данных об исполнителе задачи.

## 2.2 Проектирование локальной базы данных

Локальная база данных используется для обеспечения автономности. Пользователь, даже не имя доступа в интернет может просмотреть текущие проекты и его задачи, создать новый проект, изменить состояние текущего проекта: добавить новую задачу или подзадачу, внести изменение в состояние текущих задач. Локальная база данных имеет аналогичные глобальной структуру и связи, однако с целью реализации обратной синхронизации были добавлены некоторые столбцы в таблицы «AppUser», «Project», «ProjectMember», «SubTask», «SubTaskExecutor». Данные столбцы можно увидеть в схеме локальной базы данных, которая продемонсторирована на рисунке 2.2.

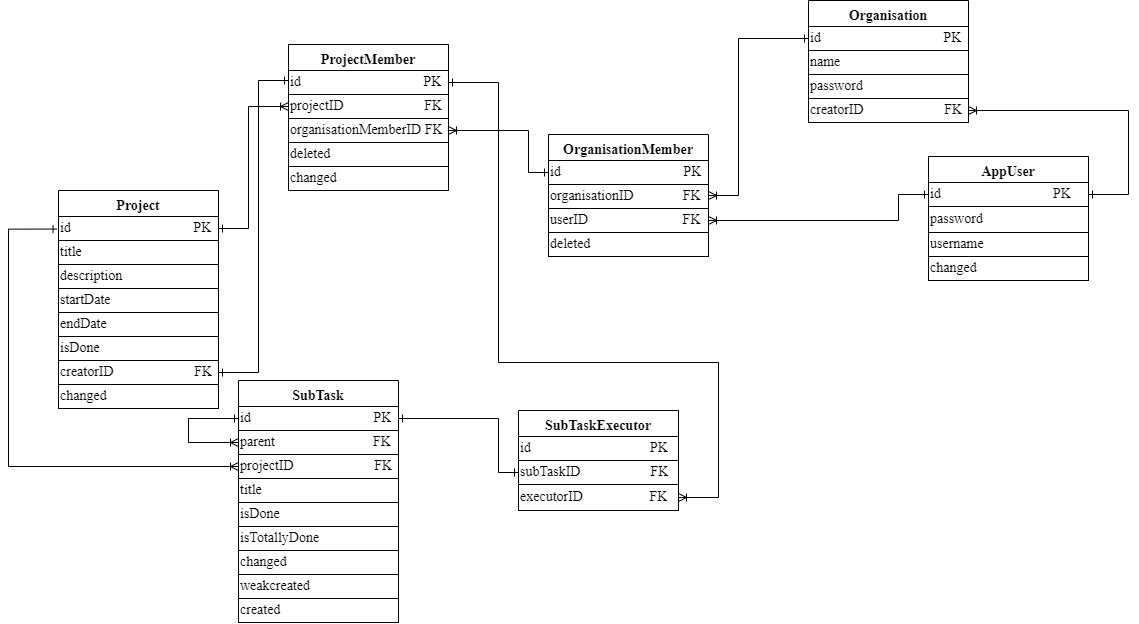


Рисунок 2.2 – схема локальной базы данных

Столбцы changed в таблицах «AppUser», «Project», «ProjectMember», «SubTaskExecutor» и «SubTask» используются для идентификации записей, которые были внесены в локальную базу данных, однако на данный момент они не существуют в глобальной базе данных.

## 2.3 Проектирование сервисов доступа к данным

Для синхронизации двух и более клиентов необходимо серверное приложение. Серверная часть приложения должна предоставлять WEB API для доступа к каждой сущности в базе данных. Передаваемые или получаемые данные от сервера должны передаваться в формате JSON.

Серверное приложение должно быть построено на основе архитектурного стиля REST и будет использоваться для взаимодействия компонентов распределённого приложения в сети.

Сервер содержит в себе реализованные сервисы для взаимодействия с каждой сущностью базы данных.

При обращении к серверу по заданному маршруту и определенному HTTP методу сервер будет определять с какой сущностью базы данных необходимо взаимодействовать и какие действия над данными необходимо будет произвести.

При помощи HTTP метода GET сервер будет возвращать данные клиенту, ссылаясь на аргументы в запросе, как на условие для выборки данных. Метод POST в свою очередь позволяет передать данные на сервер для последующей обработки. Метод PUT позволяет обновить соответствующие данные, переданные с клиента. Метод DELETE удаляет с базы данных заданную запись.

## 2.4 Проектирование мобильного приложения

Приложение должно обеспечивать пользователю осуществлять планирование и выполнение проектов.

На данном этапе был разработан макет приложения. Макет экранов входа, регистрации и экрана со списком проектов представлены на рисунке 2.3.

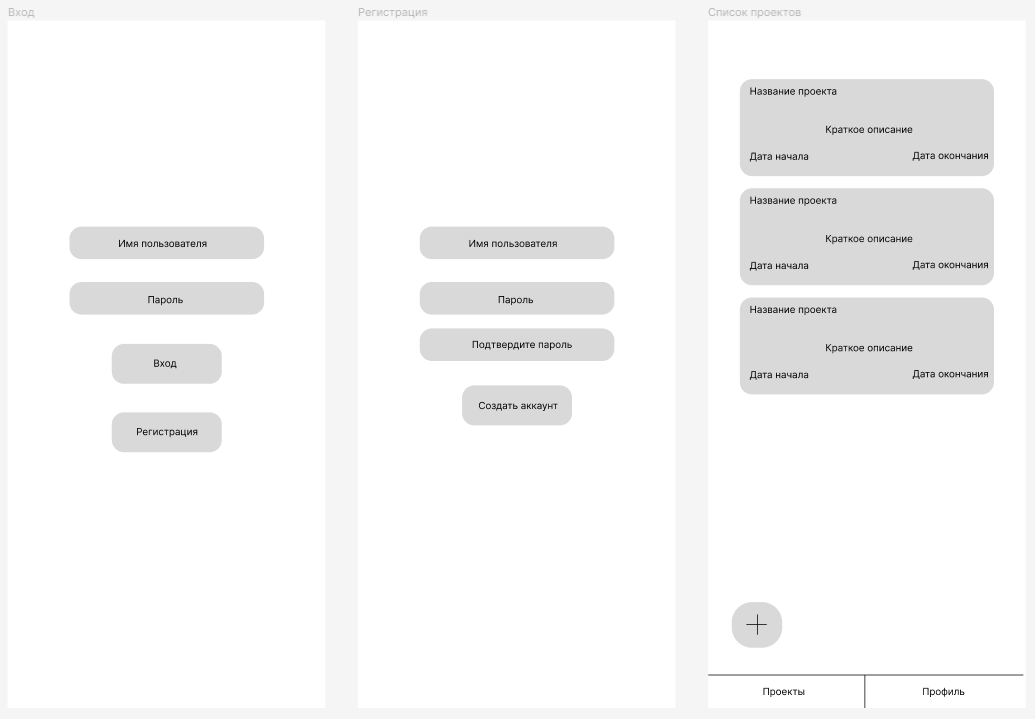


Рисунок 2.3 – экраны вход регистрация и список проектов

Макет экранов Подробности о проекте Создание нового проекта и создание подзадача представлены на рисунке 2.4.

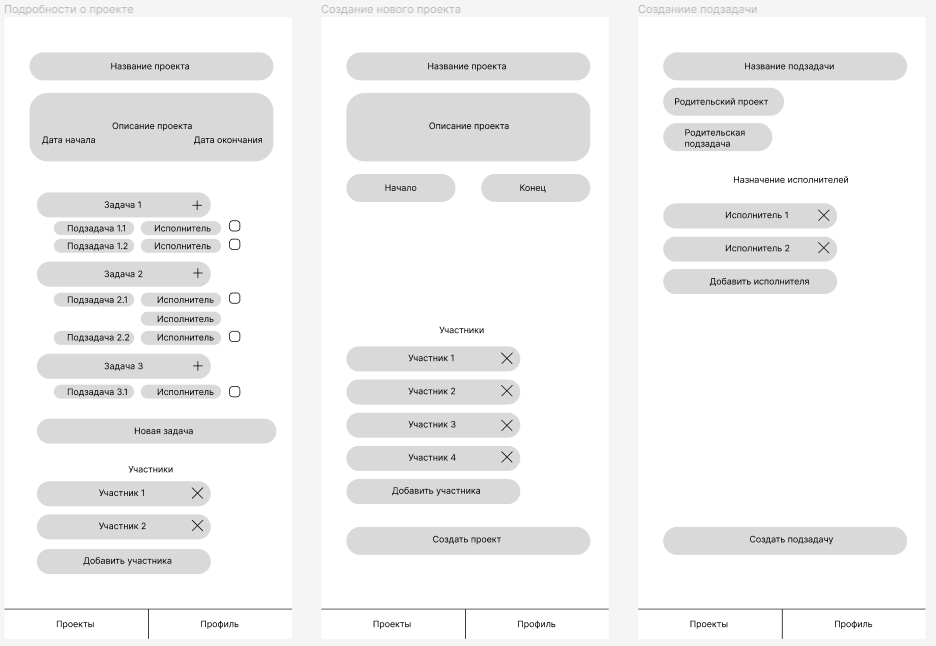


Рисунок 2.4 – экраны вход регистрация и список проектов

Макет экранов создание новой организации добавление участника и управление организацией представлены на рисунке 2.5.



Рисунок 2.5 – экраны вход регистрация и список проектов

Макет экранов Профиль и вступление в организацию представлены на рисунке 2.6.

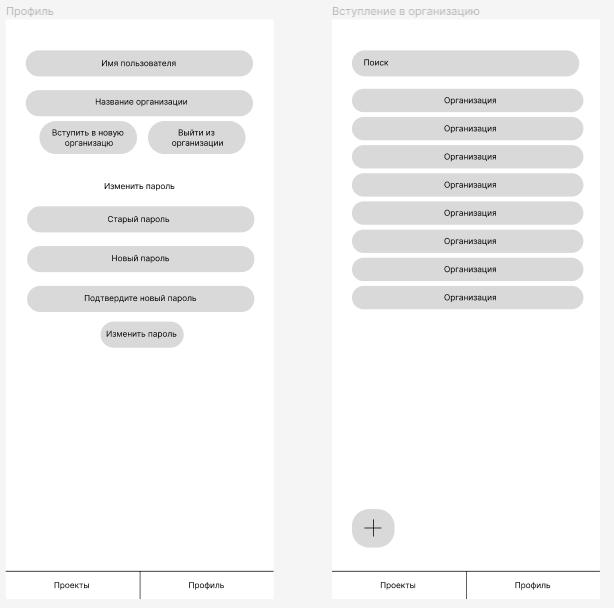


Рисунок 2.6 – экраны профиль и вступление в организацию

В ходе разработки были приняты некоторые решения по изменению дизайна конечного приложения: изменение положения некоторых элементов с целью экономии пространства и преобразования экранов добавления подзадачи и добавления участника в формат диалогового окна в связи с избыточностью создания отдельных экранов для выполнения данных действий.

Также на этапе проектирования мобильного приложения была разработана диаграмма вариантов использования. Целью разработки диаграммы является описание функционала мобильного приложения, который будет доступен каждой группе пользователей. Диаграмма представлена на рисунке 2.7.

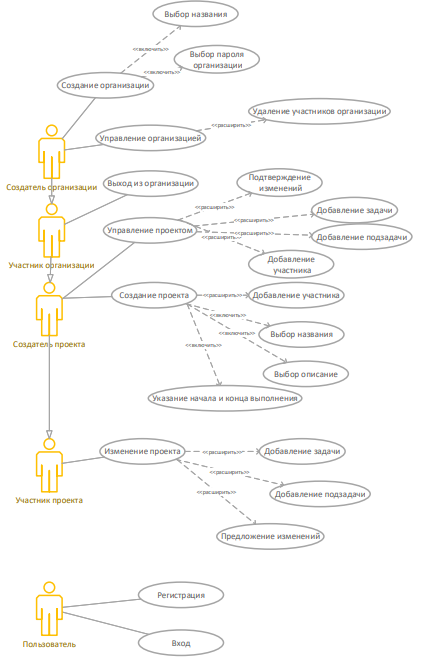


Рисунок 2.7 – диаграмма вариантов использования

Согласно диаграмме, в приложении должны быть реализованы следующие функции:

* регистрация пользователя;
* авторизация пользователя;
* создание организации;
* создание проекта;
* вступление в организацию;
* добавление подзадачи;
* добавление подзадачи;
* изменение состояния подзадачи.

## 2.5 Проектирование веб приложения

На этапе проектирования веб приложения было принято решение о разработке веб приложения для реализации функций админа: удаление подзадачи, удаление задачи, удаление проекта и так далее.

Был разработан макет для веб приложения, который, который состоит из трех экранов: экран входа, список проектов, экран проекта.

Экран входа представлен на рисунке 2.8



Рисунок 2.8 – экран входа веб приложения

Экран со список проектов представлен на рисунке 2.9.



Рисунок 2.9 – экран со списком проектов

Экран с информацией о конкретном проекте на рисунке 2.10.

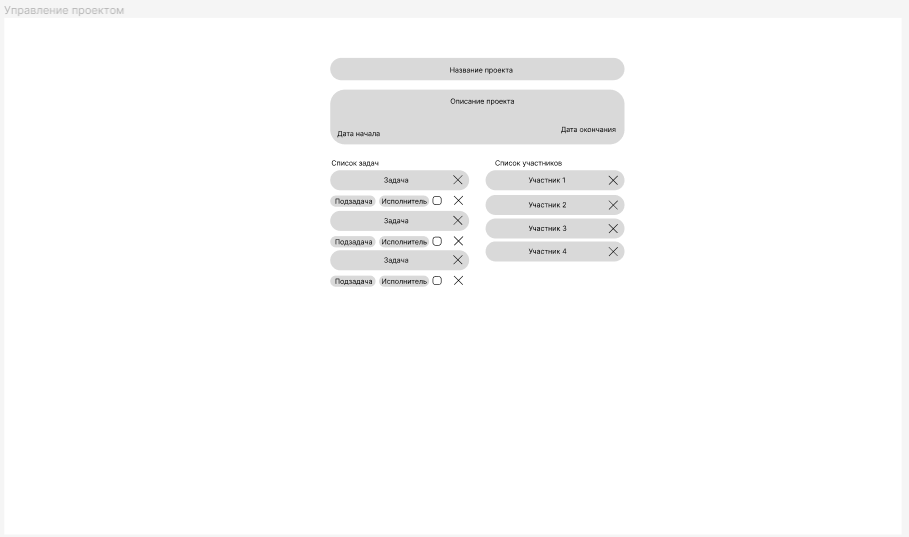


Рисунок 2.10 – экран управления проектом

Для веб приложения также была разработана диаграмма вариантов использования. Данная диаграмма отображает функционал веб приложения, доступный создателям проекта. Диаграмма вариантов использования веб приложения представлена на рисунке 2.11.

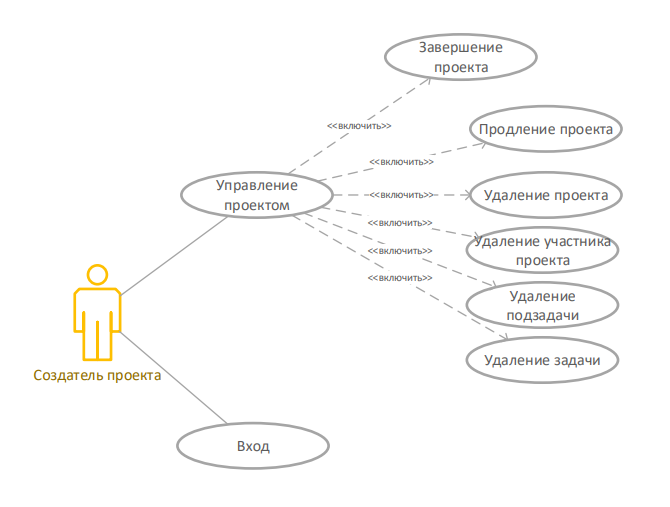


Рисунок 2.11 – диаграмма вариантов использования веб приложения

## 2.6 Выводы по главе «Проектирование программного продукта»

В данной главе было рассмотрено проектирование каждого отдельного компонента приложения.

Были созданы макеты для мобильного и веб приложений. Также были разработаны диаграммы вариантов использования для определения основного функционала, доступного каждой из групп пользователей.

При проектировании баз данных были разработаны таблицы, столбцы и связи между таблицами для локальной и глобальной баз данных.

В ходе проектирования сервера был выбран формат передачи данных, а также основные методы, которые будут использованы для передачи.

# **3 Программная реализация продукта**

## 3.1 Технические средства разработки

Для разработки программного продукта были использованы следующие средства: Microsoft SQL Server, Flutter, SQLite, Node.js и Express.

Microsoft SQL Server – система управления реляционными базами данных, разработанная корпорацией Microsoft. Microsoft SQL Server была использована для разработки глобальной базы данных.

Flutter – комплект средств разработки и фреймворк для создания мобильных приложений под Android и iOS, веб приложений, а также настольных приложений с использованием языка программирования Dart. Flutter был использован при разработке мобильного и веб приложений.

Flutter был выбран за счет своей удобности и простоты использования.

SQLite – это встраиваемая кроссплатформенная БД, которая поддерживает достаточно полный набор команд SQL и доступна в исходных кодах.

Node.js – среда выполнения кода на JavaScript, которая построена на основе движка JavaScript Chrome V8, который позволяет транслировать вызовы на языке JavaScript в машинный код. Node.js прежде всего предназначен для создания серверных приложений на языке JavaScript.

Express – это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений.

В качестве среды разработки мобильного и веб приложений была выбрана Android Studio – интегрированная среда разработки для работы с платформой Android.

Для разработки сервера же был выбран Visual Studio Code – текстовый редактор, разработанный Microsoft.

## 3.2 Разработка глобальной базы данных

Как уже упоминалось, для разработки глобальной базы данных использовалась СУБД Microsoft SQL Server.

Скрипт для создания глобальный базы данных представлен в приложении А.

Все CRUD операции базы данных проводятся с помощью хранимых процедур. Некоторые из хранимых процедур представлены на рисунке 3.1.

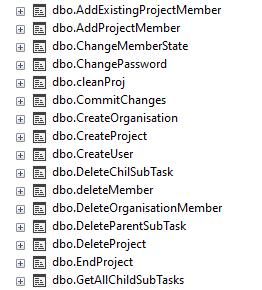


Рисунок 3.1 – некоторые из хранимых процедур глобальной базы данных

## 3.3 Разработка локальной базы данных

Локальная база данных полностью дублирует структуру глобальной базы данных. Для взаимодействия с локальной базой данных была использована библиотека sqflite – библиотека для взаимодействия с sqlite – базами данных.

Скрипт для создания локальной базы данных представлен в приложении Б.

## 3.4 Разработка сервера

Для разработки сервера был выбран программная платформа Node.js и фреймворк Express. Совокупность данных средств позволяет с легкостью разработать сервер для обработки http-запросов.

Для обработки запросов используются определенный ряд встроенных функций, таких, как например функция get, представленная в листинге 1.

app.get("/organisation/getAllOrganisations",function(req,response){

    const request = new sql.Request();

    request.execute("GetAllOrganisations",(err,result)=>{

        response.end(JSON.stringify(result.recordset));

    })

})

Листинг 1 – пример обработки GET-запроса

В качестве параметров для этой функции передаются маршрут и обработчик, который будет вызываться, если запрос к серверу соответствует данному маршруту. Таким же образом выполняется обработка POST – запросов. Пример обработки POST – запроса представлен в листинге 2.

app.post("/project/addSubTask",function(req,response){

    console.log("proejcts call");

    request = new sql.Request();

    request.input('userID',req.query.userID);

    request.execute('GetAllUserProjects',(err,result)=>{

        response.statusCode=200

        response.end(JSON.stringify(result.recordset));

    })

})

Листинг 2 – пример обработки POST – запроса

## 3.5 Разработка мобильного приложения

При разработке мобильного приложения была использована среда разработки Android Studio. Структура проекта представлена на рисунке 3.2.

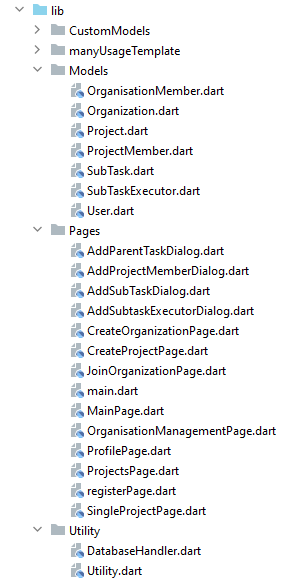


Рисунок 3.2 – структура проекта мобильного приложения

Как видно из рисунка 3.4 проект состоит из пяти основных директорий: CustomModels, manyUsageTemplates, Models, Pages, Utility.

В директории Models находятся классы, который используются для взаимодействия с базами данных. Каждый файл содержит класс, чьи поля совпадают со столбцами соответствующей таблицы.

В директории CustomModels находятся классы, которые используются для выполнения сложных запросов, в которых объединяется множество таблиц, к базам данных.

В директории manyUsageTemplates находятся виджеты, многократно используемые в проекте, например текстовое поле или кнопки.

В директории Utility расположены два файла – DatabaseHandler и Utility. Файл Utility содержит в себе класс, который обеспечивает доступ свойствам, которые должны быть доступны из любого места программы, например ip-адрес и порт сервера. Файл DatabaseHandler, как следует из названия, предназначен для взаимодействия с локальной базой данных. В данном файле определены функции для синхронизации и получения данных из локальной базы данных, происходит начальная инициализация и создание базы данных.

В директории Pages находятся файлы, которые представляют из себя страницы приложения.

Каждый класс, который представляет из себя страницу имеет метод build, который возвращает интерфейс, который будет отображен на экране пользователя.

Для взаимодействия с сервером была использована библиотека http. Данная библиотека позволяет отправлять http запросы используя язык Flutter. Пример использования библиотеки http представлен в листинге 3.

final String url = "http://${Utility.*url*}/project/addMembers";  
final response = await http.post(Uri.*parse*(url), headers: <String, String>{  
 'Content-Type': 'application/json;charset=UTF-8',  
}, body: jsonEncode(<String, String>{  
 'organisationID': member.organisationID.toString(),  
 'projectID': project.id.toString(),  
}));  
if (response.statusCode == 200) {  
 InitializeProject();  
} else {  
 print("Ошибка");  
}

Листинг 3 – пример использования библотеки http

При отсутствии подключения данные для вывода пользователю извлекаются из локальной базы данных. Для взаимодействия с локальной базы данных необходимо получить объект класса Database – класса, который предоставляет методы для взаимодействия с базой данных. Для выполнения запросов к базе данных используется метод rawQuery, в качестве параметров которого передаются запрос и его параметры. Пример получения данных из внутренней базы данных представлен в листинге 4.

Future<List<CustomProject>> getProjectsFromLocal() async{  
 final db= await database;  
 List<CustomProject> projectsBuffer = [];  
 final List<Map<String,dynamic>> maps = await db.rawQuery("select Project.id,Project.title,Project.creatorID,"  
 "Project.decription as Description,Project.endDate,Project.isDone,Project.startDate From Project "  
 "inner join ProjectMember on Project.id = ProjectMember.projectID "  
 "inner join OrganisationMember on OrganisationMember.id = ProjectMember.organisationMemberID "  
 "where OrganisationMember.userID=? and OrganisationMember.deleted=0",[Utility.*user*.id]);  
 maps.toList().forEach((element) {  
 print("iteration");  
 int valueBuffer = element.putIfAbsent("isDone", () => null);  
 bool booleanBuffer;  
 if(valueBuffer == 1){  
 booleanBuffer=true;  
 }else{  
 booleanBuffer=false;  
 }  
 projectsBuffer.add(CustomProject(element.putIfAbsent("id", () => null), element.putIfAbsent("title", () => null),  
 element.putIfAbsent("Description", () => null), element.putIfAbsent("startDate", () => null), element.putIfAbsent("endDate", () => null),  
 booleanBuffer, element.putIfAbsent("creatorID", () => null)));  
 });  
 return projectsBuffer;  
}

Листинг 4 – получение списка проектов из локальный базы данных

Для реализации синхронизации был разработан ряд методов в классе DatabaseHandler и ряд отдельных обработчиков маршрутов на сервере. При входе пользователя в приложение производится синхронизация – используя идентификатор пользователя загружается вся информация, связанная с данным пользователем: пароль, организация, проекты пользователя, список задач и участников проекта. Список методов продемонстрирован в таблице 3.1.

**Таблица 3.1 – Список методов синхронизации**

|  |  |
| --- | --- |
| getAllData() | Главный метод, вызывающий дочерние |
| getOrg() | Получение информации об организации |
| getProjMembers() | Получение списка участников проектов |
| getProjects() | Получение списка проектов |
| getSubTasks() | Получение списка задач и подзадач |
| getSubTasksExecutors() | Получение списка исполнителей подзадач |

Также в приложении была реализована обратная синхронизация: в случае если у пользователя отсутствует доступ в сеть, он может внести некоторые изменения, такие как создание проекта, создание задачи или подзадачи, изменение состояния задачи, а после подключения к сети происходит выгрузка внесенных изменений в глобальную базу данных.

Для того чтобы определить какие из записей базы данных были изменены, в локальной базе данных существуют дополнительные столбцы, упомянутые в разделе «Проектирование локальной базы данных».

При запуске приложения вызывается метод uploadData, который в свою очередь вызывает ряд методов, осуществляющих обратную синхронизацию. Метод uploadData представлен в листинге 5.

Future<void> uploadData()async{  
 await uploadAppUserData();  
 await uploadProjectsData();  
 uploadUnboundedParentSubTasks();  
 uploadWeakCreatedChildSubTasks();  
 uploadSimplyChangedChildSubTasks();  
}

Листинг 5 – метод uploadData

## 3.7 Разработка веб приложения

Для разработки веб приложения также был использован фреймворк Flutter, так как он позволяет разрабатывать приложения для запуска в браузере.

Веб приложение является по сути панелью администратора, которая предоставляет возможность управления проектом. Структура веб приложения представлена на рисунке 3.3.

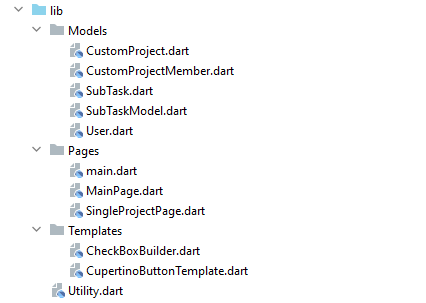


Рисунок 3.3 – структура проекта веб приложения

Все директории имеют назначение, аналогичное назначению соответствующих директорий в проекте мобильного приложения.

Для взаимодействия с сервером также используется библиотека http.

Кроме возможности управления проектом веб приложение предоставляет статистику по проектам в графическом виде. Для графического представления прогресса используется библиотека flutter\_animation\_progress\_bar. Используя веб приложение можно увидеть количество выполненных подзадач, а также количество прошедшего времени в процентном соотношении.

Для взаимодействия с проектами используется библиотека context\_menus. Данная библиотека позволяет создавать пользовательские контекстные меню для элементов страницы. Код для создания пользовательского контекстного меню представлен в листинге 6.

ContextMenuRegion(  
 contextMenu: GenericContextMenu(  
 buttonConfigs: [  
 ContextMenuButtonConfig("Продлить дату проекта", onPressed: ()async{  
 setDate(index);  
 }),  
 ContextMenuButtonConfig("Завершить проект", onPressed: (){  
 endProject(projects[index]);  
 print("end click");  
 }),  
 ContextMenuButtonConfig("Удалить проект", onPressed: (){  
 deleteProject(projects[index]);  
 print("delete click");  
 })  
 ],  
 ),  
 child: Card(  
 elevation: 10,  
 shape: RoundedRectangleBorder(  
 side: const BorderSide(  
 color: Colors.*blue*,  
 ),  
 borderRadius: BorderRadius.circular(100)  
 ),  
 shadowColor: Colors.*blue*,  
 child: inkWell(index);  
 ),  
),

Листинг 6 – код создания пользовательского контекстного меню

Одной из возможностей управления проектом является изменения даты завершения проекта. С целью свести к минимуму возможность возникновения ошибки во время ввода пользователем значения было принято решение предоставить пользователю способ выбора новой даты без необходимости прибегать к вводу с клавиатуры.

Для реализации данной задумки была использована библиотека syncfusion\_flutter\_datepicker. Пример использования данной библиотеки представлен в листинге 7.

SfDateRangePicker(  
 onSelectionChanged: (DateRangePickerSelectionChangedArgs args){  
 setState(() {  
 newEndDate = args.value.toString().substring(0,10);  
 print(newEndDate);  
 });  
 },  
 selectionMode: DateRangePickerSelectionMode.single,  
 initialSelectedDate: DateTime.now(),  
),

Листинг 7 – использование библиотеки syncfusion\_flutter\_datepicker

В листинге 7 приводится создание объекта, который представляет из себя календарь, с помощью которого пользователь сможет выбрать желаемую дату.

## 3.8 Выводы по главе «Программная реализация продукта»

В данной главе была рассмотрена разработка каждого из компонентов программного продукта.

Сперва был рассмотрен спектр технических средств, используемых при разработке приложения. Затем была рассмотрена разработка локальной и глобальной баз данных

Третий этап – разработка веб сервера с использованием Node.js

На четвертом этапе была рассмотрена разработка мобильного приложения

Заключительным этапом главы стала разработка полноценного веб приложения, используемого в качестве панели администратора.

# **4 Обеспечение безопасности информационной системы**

Для обеспечения безопасности данных и предотвращения несанкционированного доступа к аккаунтам пользователей необходимо обеспечить хеширование паролей.

Для этого была использована библиотека crypto, предоставляющая набор хеширующих функций для языка Dart. В состав библиотеки входят алгоритмы SHA-1, SHA-224, SHA-256, MD5.

Для данного приложения был выбран алгоритм MD5 – 128-битный алгоритм хеширования, разработанный профессором Рональдом Ливестом из Массачусетского технологического института.

При этом хеширование должно осуществляться на стороне мобильного приложения, так как в ходе передачи пароля на сервер, он может быть перехвачен злоумышленниками.

## 4.1 Вывод по главе «Обеспечение безопасности информационной системы»

В данной главе было рассмотрено обеспечение безопасности информационной системы проекта.

Так как в данном курсовом проекте присутствует авторизация пользователя, то было решено использовать хеширование паролей в качестве обеспечения информационной безопасности системы.

# **5. Руководство пользователя**

## 5.1 Руководство пользователя мобильного приложения

При входе в приложение пользователь попадает на экран входа (рис.5.1).

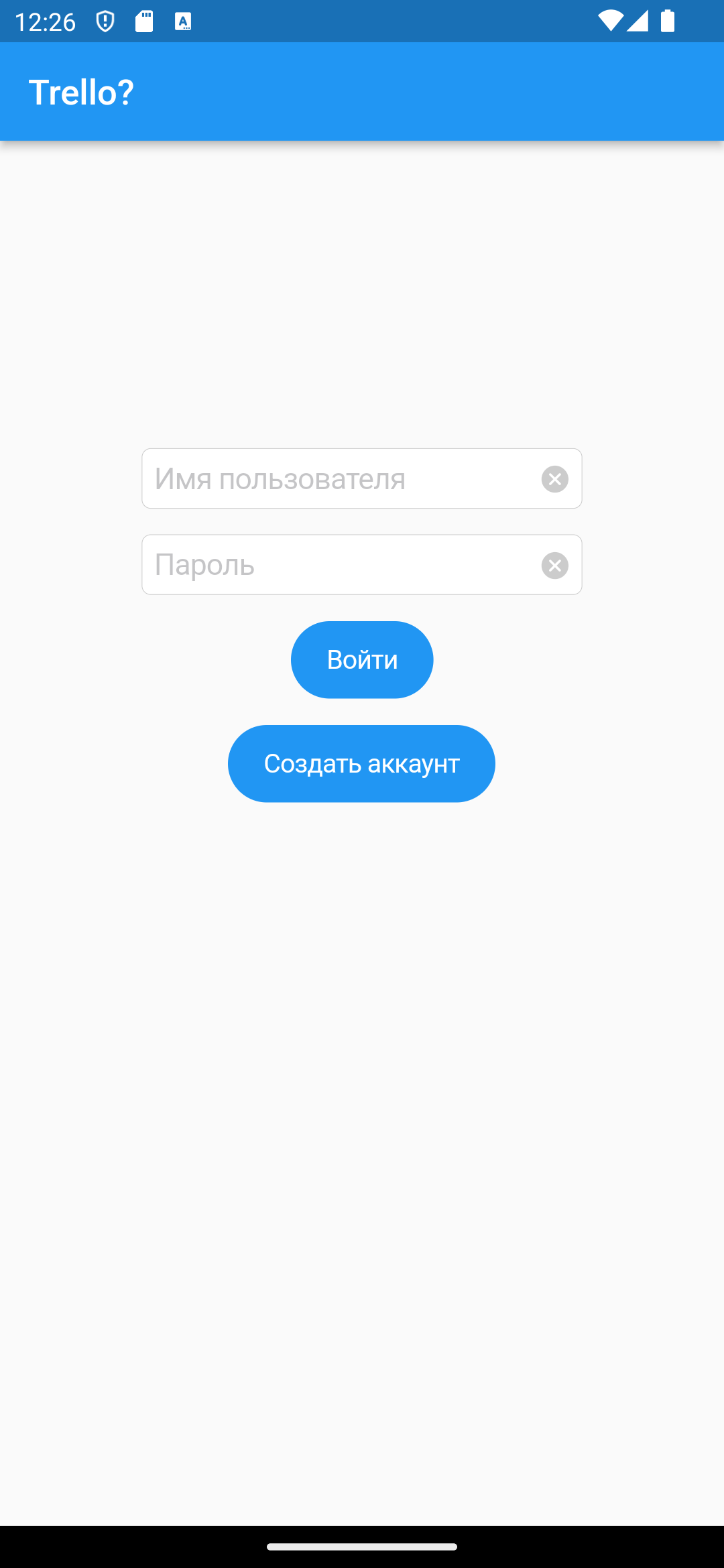


Рисунок 5.1 – страница «Вход» мобильного приложения

Сперва пользователю необходимо нажать на кнопку «Создать аккаунт», после чего будет совершен переход на страницу «Регистрация» (рис. 5.2).

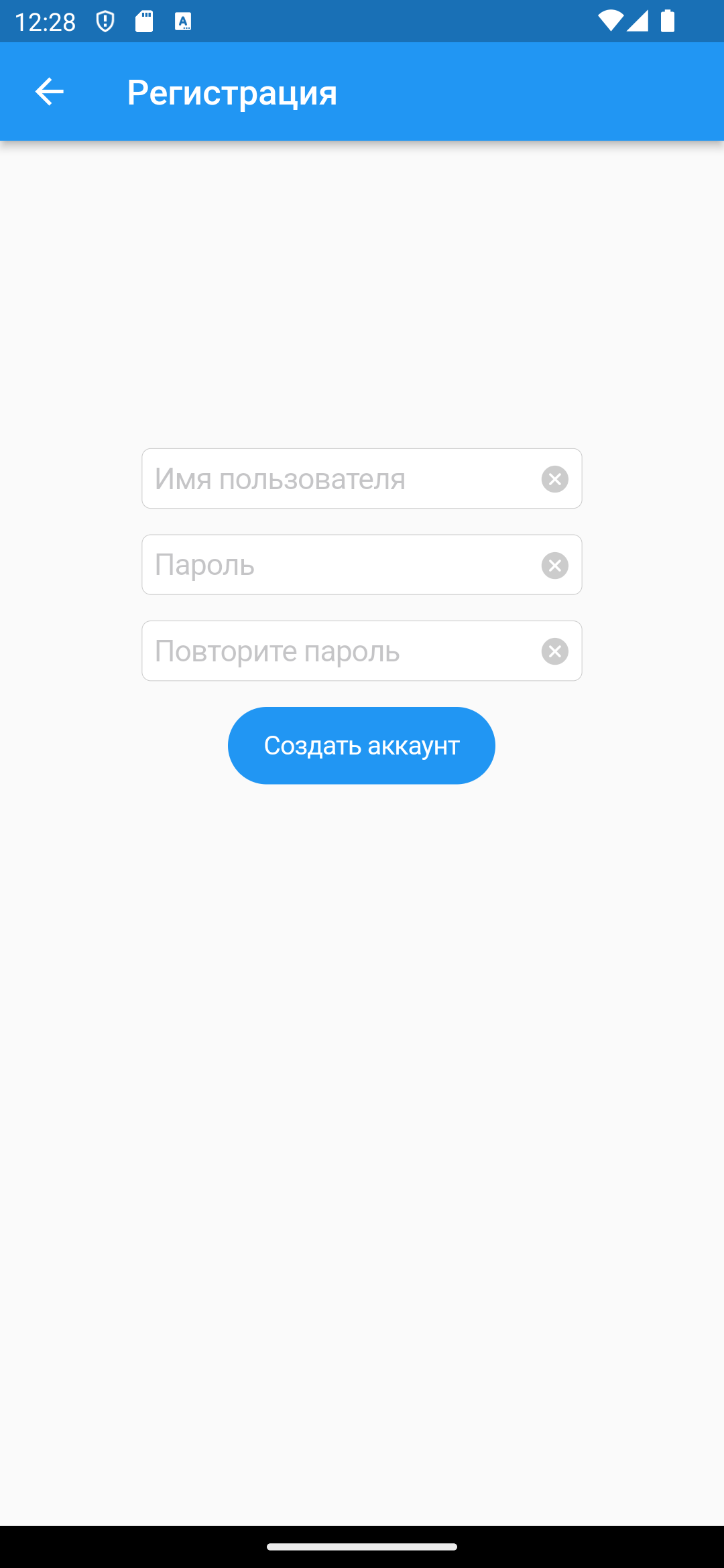


Рисунок 5.2 – страница «Регистрация» мобильного приложения

После регистрации пользователь возвращается на страницу входа и вводит свои данные, после чего нажимает на кнопку «Войти» и переходит на первую из двух главных страниц – страницу «Все проекты» (рис.5.3).



Рисунок 5.3 – страницы «Все проекты» мобильного приложения

На странице «Все проекты» пользователь видит список текущих проектов, для создания нового проекта пользователю необходимо нажать на кнопку «Добавить», расположенную в нижнем левом углу. После нажатия на кнопку добавить пользователь перенесется на страницу «Создание проекта» (рис 5.4).

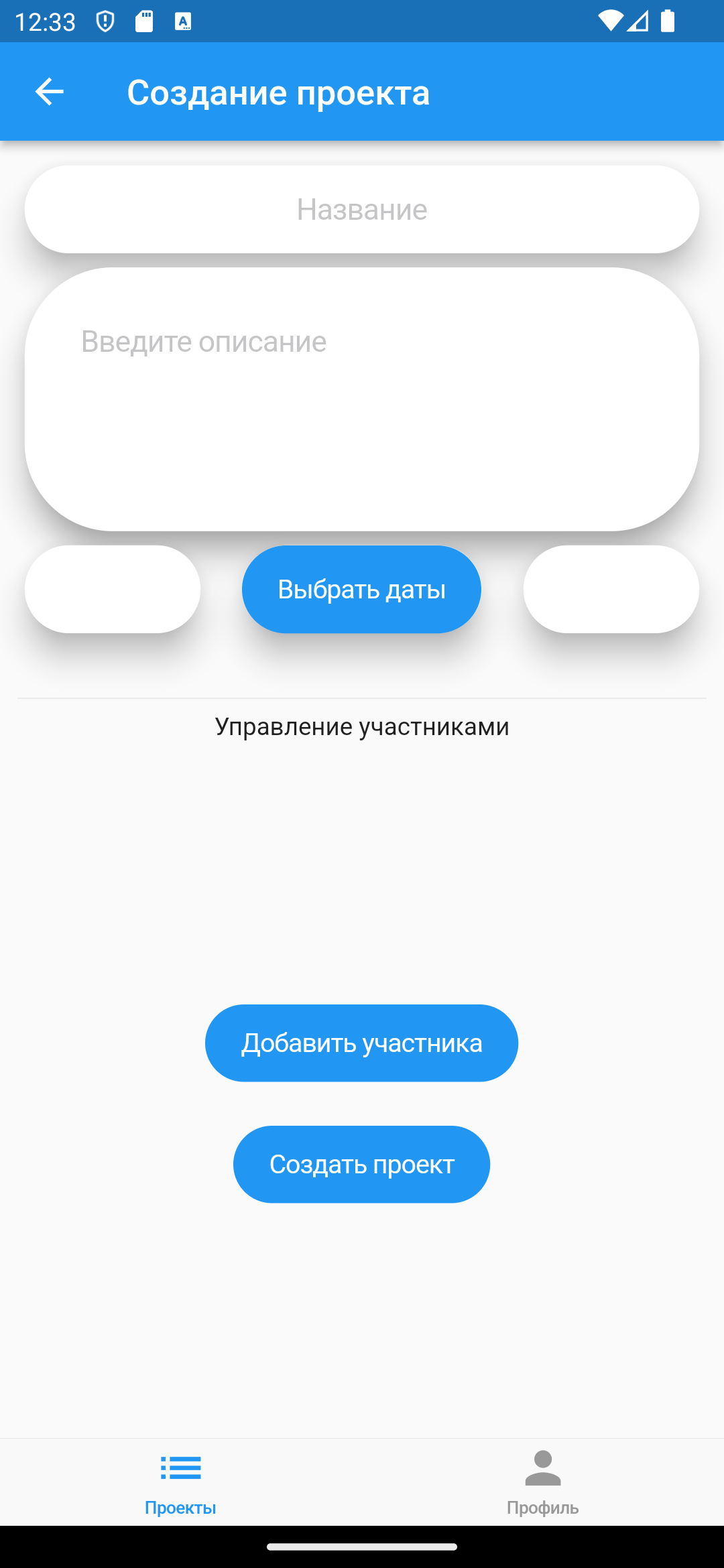


Рисунок 5.4 – страница «Создание проекта»

На данном экране пользователь вводит название и описание проекта, выбирает даты начала и конца и добавляет участников в проект. После ввода информации пользователь нажимает на кнопку «Создать проект» и возвращается на страницу «Проекты».

Для того чтобы открыть страницу конкретного проекта пользователю необходимо нажать на проект, который он хочет открыть, после этого пользователь переходит на страницу конкретного проекта (рис. 5.5).

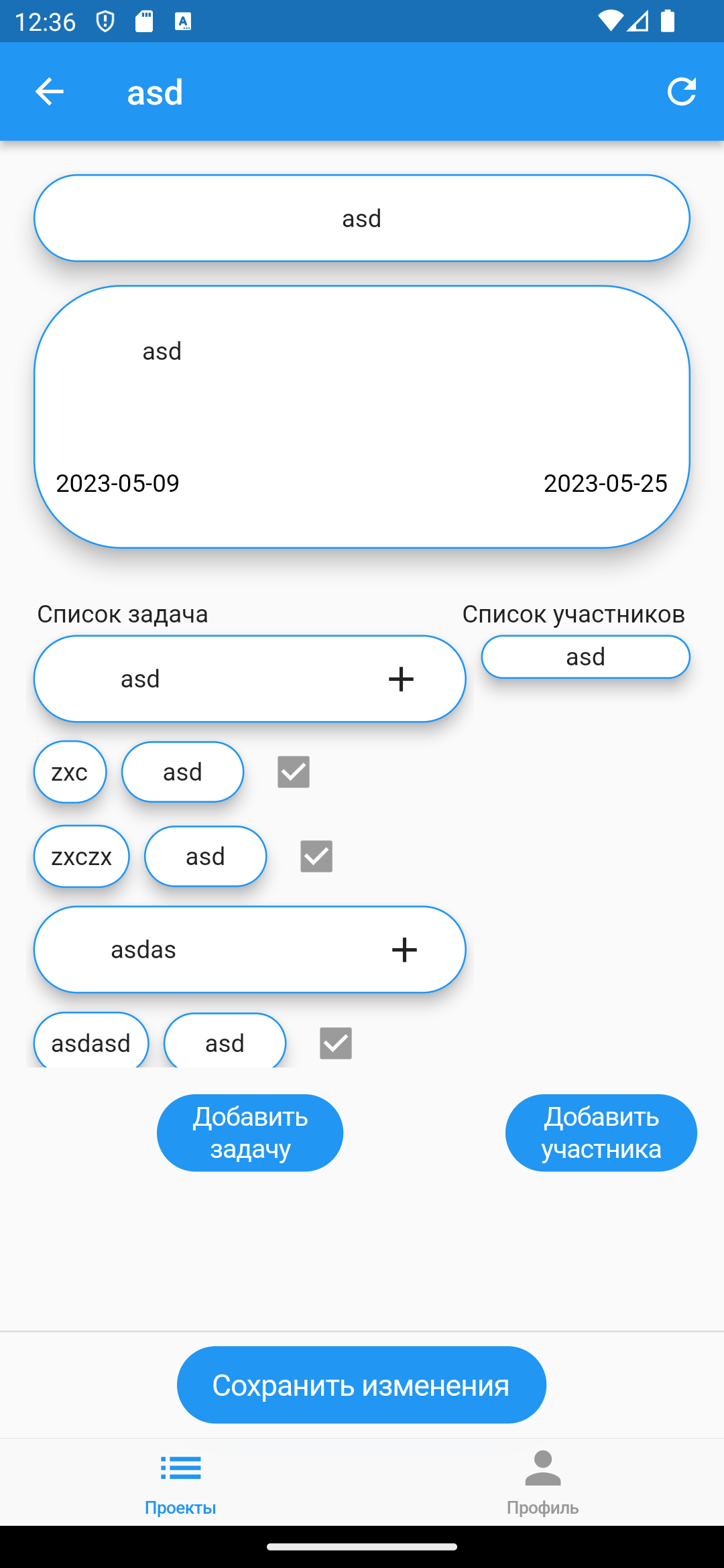


Рисунок 5.5 – страница конкретного проекта

На данной странице пользователь может добавить участника, нажав на кнопку «Добавить участника» и выбрав в диалоговом окне участника организации (рис. 5.6).

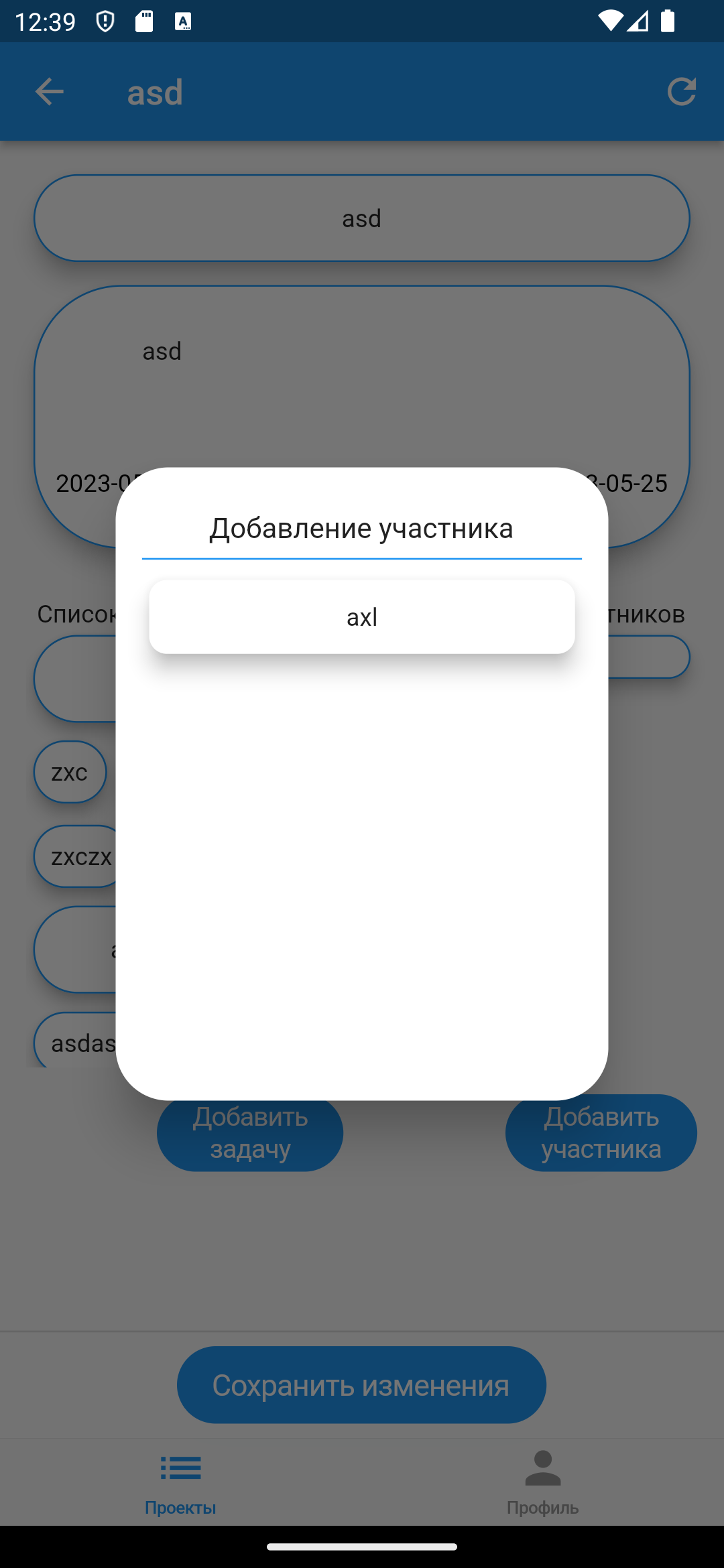


Рисунок 5.6 – добавление участника в проект

Для добавления задачи пользователю необходимо нажать на кнопку «Добавить задачу», после чего откроется диалоговое окно, в которое пользователю необходимо ввести имя задачи (рис 5.7).

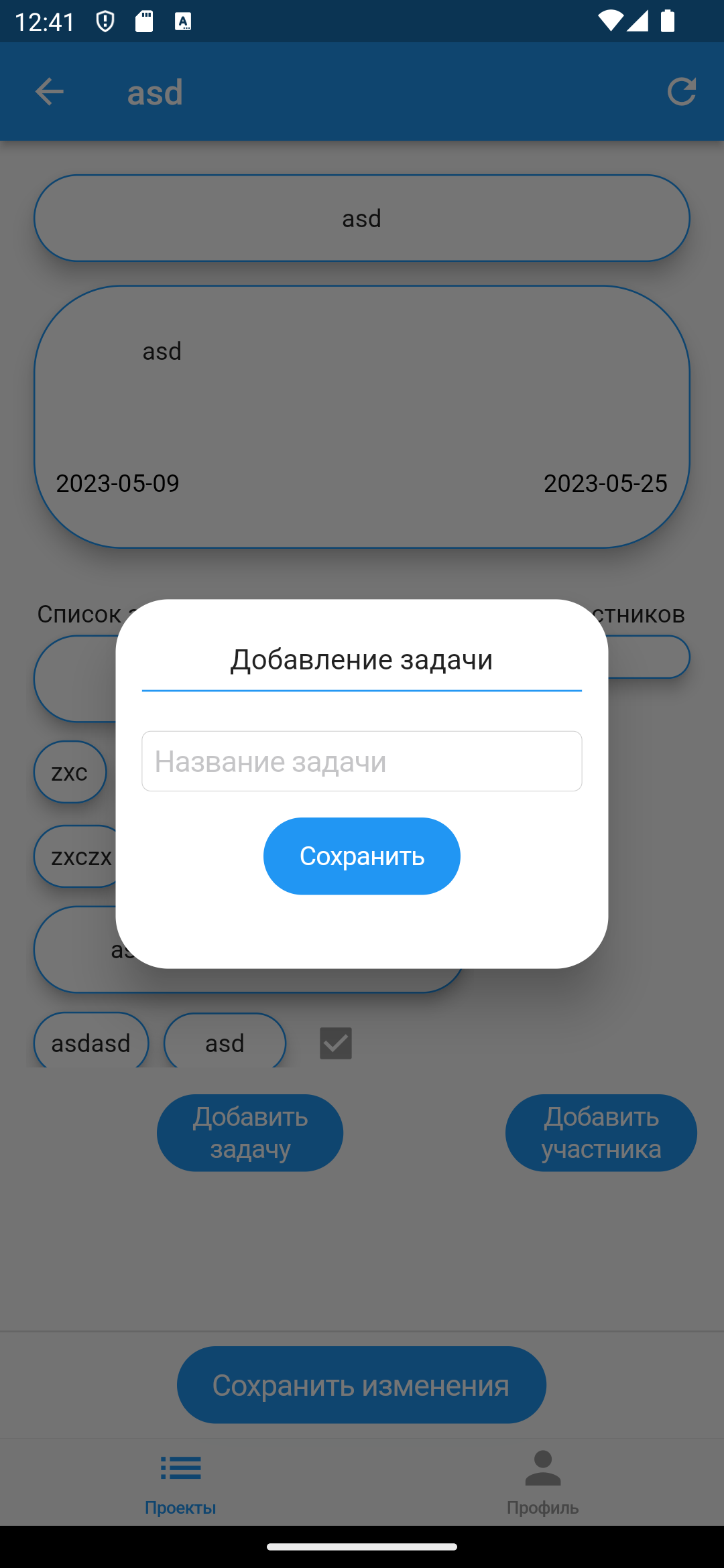


Рисунок 5.7 – добавление задачи в проект

Для добавления подзадачи пользователю необходимо на кнопку «Добавить подзадачу», которая является родительской для добавляемой подзадачи, после чего откроется диалоговое окно для добавления подзадачи. (рис. 5.8).

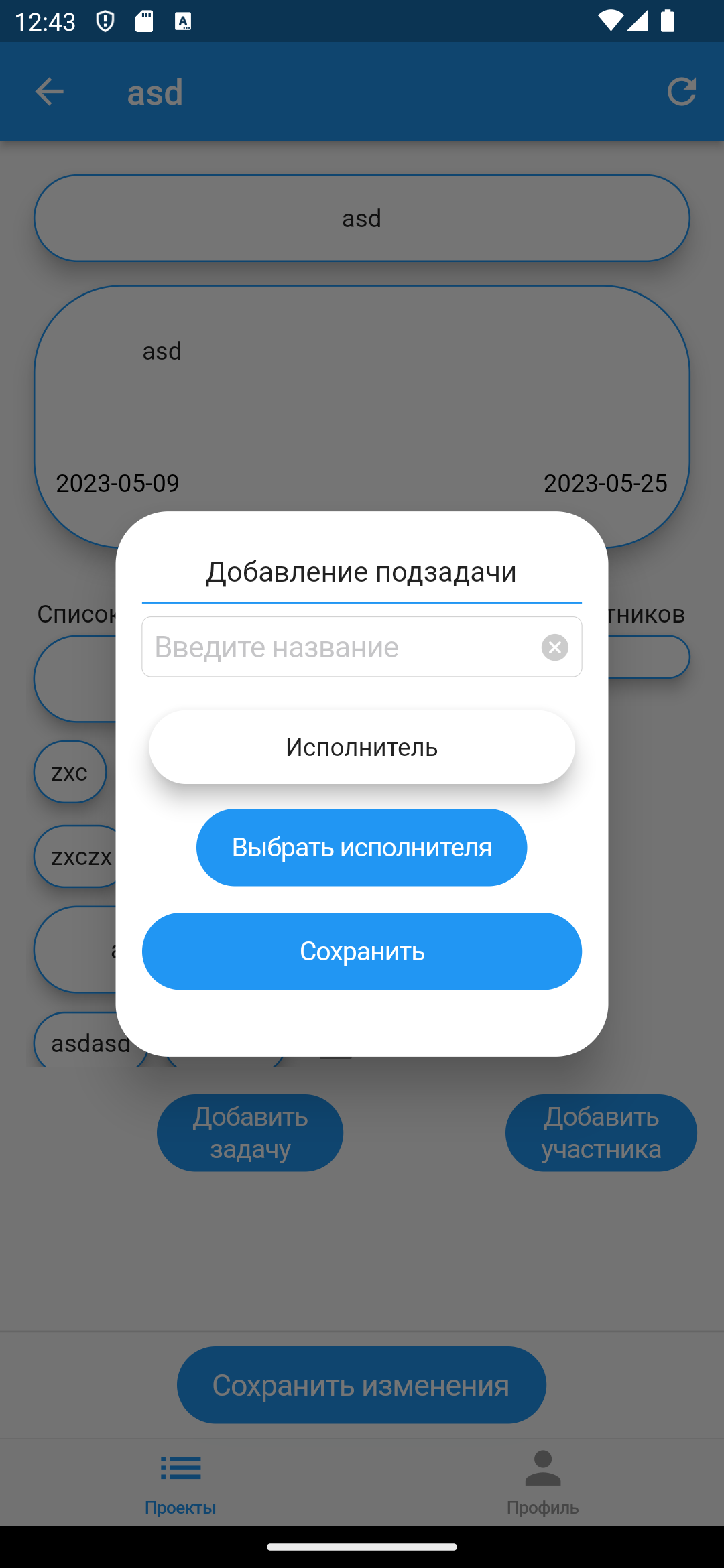


Рисунок 5.8 – добавление подзадачи в проект

В данном окне пользователю необходимо ввести название подзадачи и выбрать её исполнителя, после чего нажать на кнопку «Сохранить».

Второй из главных страниц является страница «Профиль» (рис. 5.9).

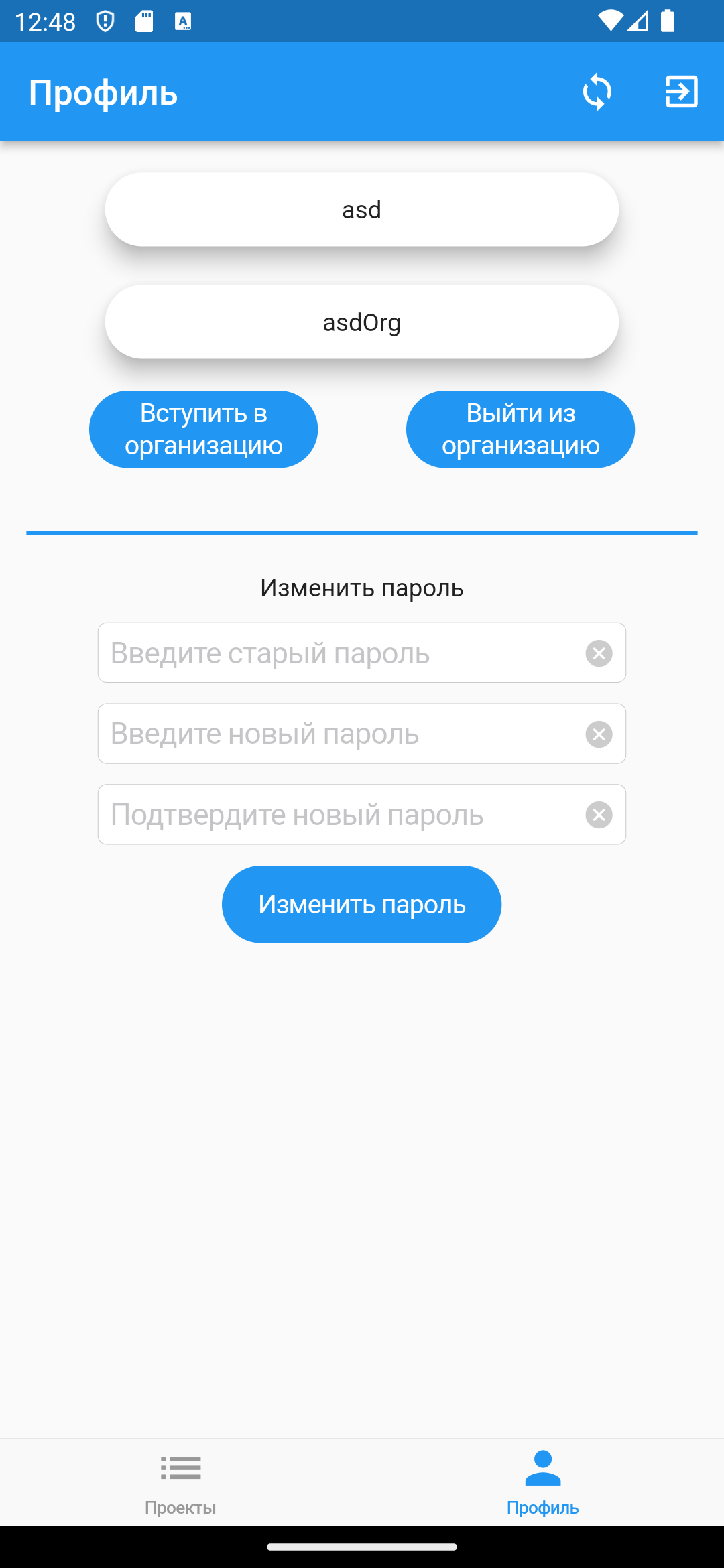


Рисунок 5.9 – страница «Профиль»

На странице «Профиль» пользователь имеет возможность сменить пароль: после заполнения необходимых полей пользователь нажимает на кнопку «Изменить пароль». При нажатии на кнопку «Выйти», расположенную в верхнем правом углу, пользователь выйдет из аккаунта. При нажатии на кнопку «Синхронизация», расположенную слева от кнопки «Выйти», будет запущен процесс синхронизации локальной и глобальной баз данных.

При нажатии на кнопку «Вступить в организацию», если пользователь на данный момент не состоит в организации, будет совершен переход на страницу «Вступить в организацию» (рис. 5.10).



Рисунок 5.10 – страница «Вступить в организацию»

На странице «Вступить» в организацию пользователь может создать свою организацию, нажав на кнопку добавить, расположенную в нижнем левом углу, после чего пользователь перейдет на страницу «Создание организации» (рис. 5.11).

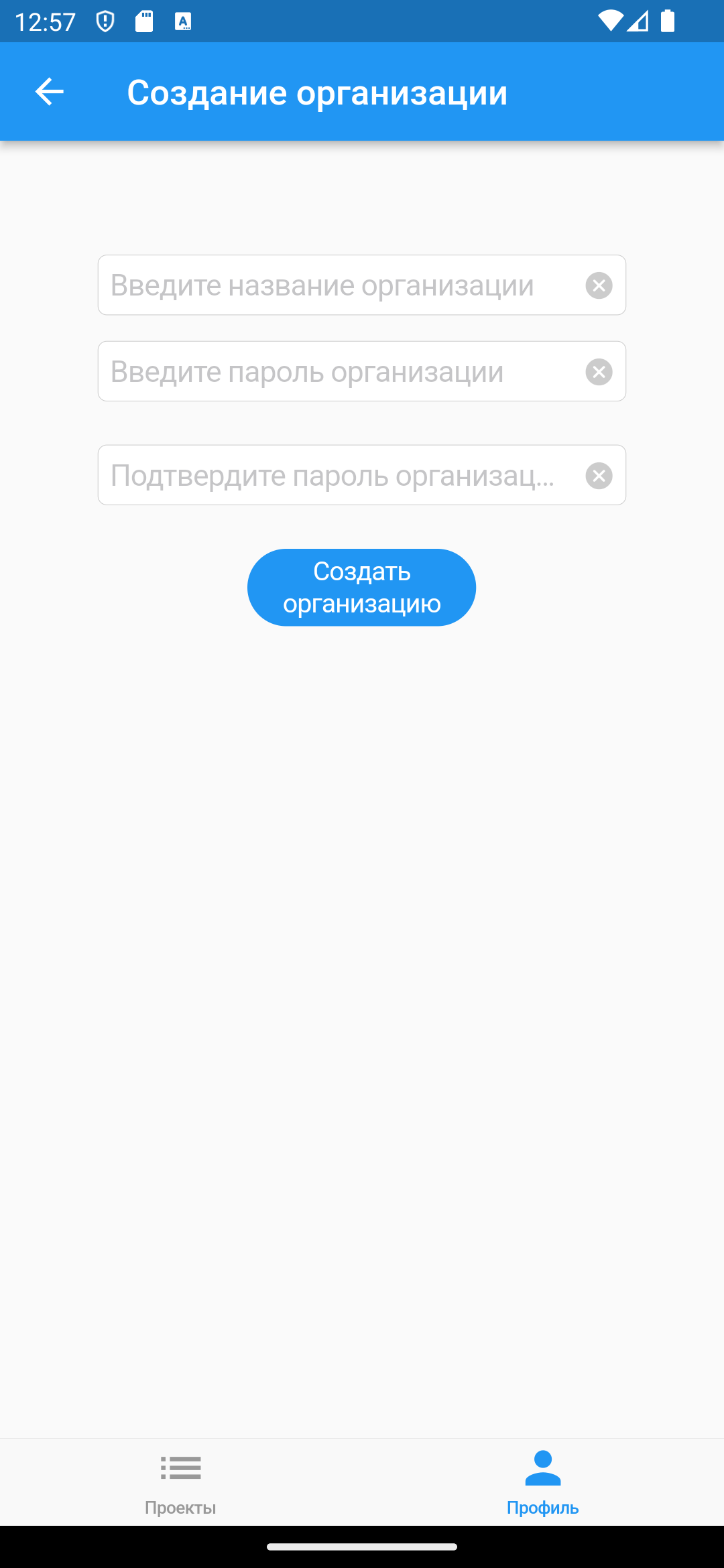


Рисунок 5.11 – страница «Создание организацию»

На данной странице пользователь должен ввести необходимую информацию и нажать на кнопку «Создать организацию», после чего организация будет создана.

Также на странице «Профиль» пользователь может открыть страницу управления организацией, при нажатии на элемент с названием названия организации, в том случае, если пользователь является создателем организации, пользователь перейдет на страницу управления организацией (рис. 5.12).

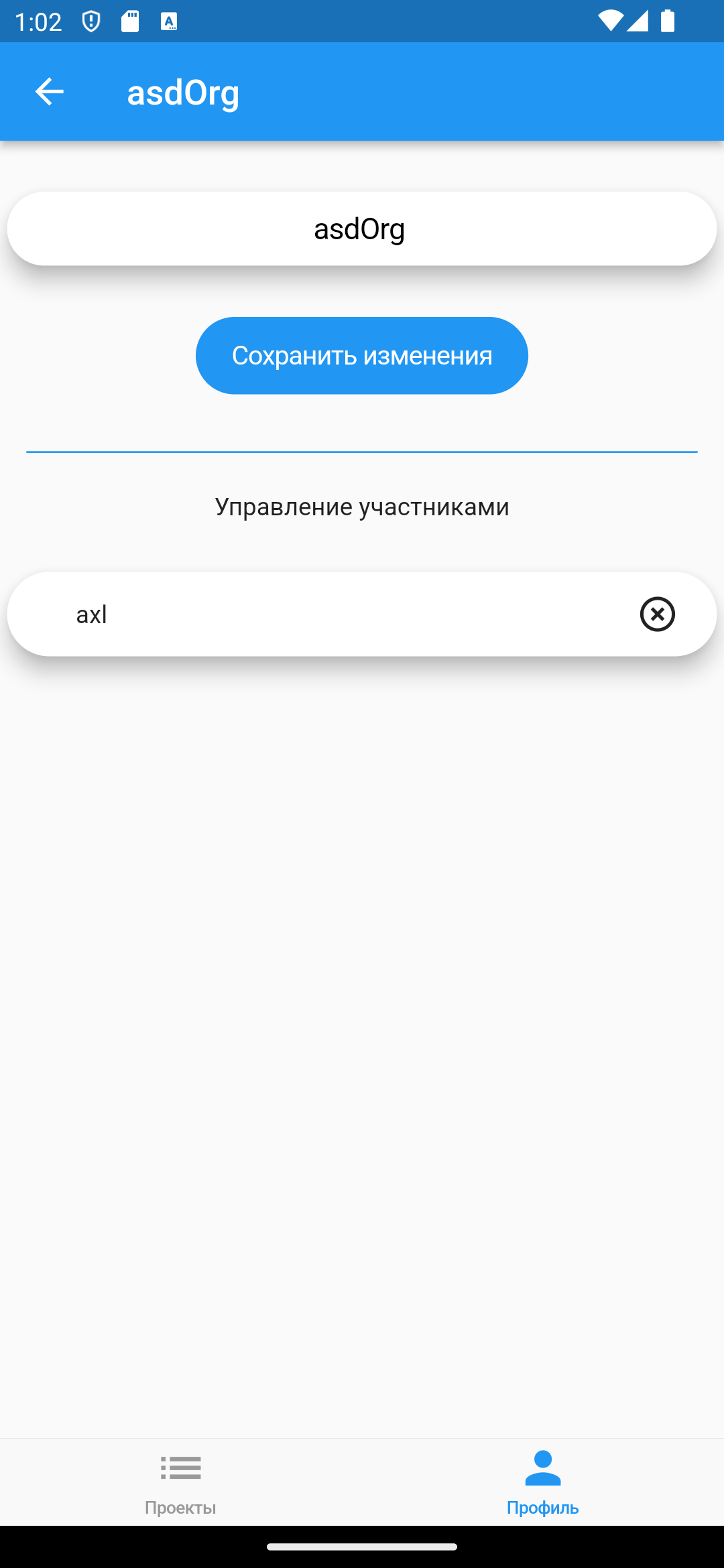


Рисунок 5.12 – страница управления организацией

На данной странице пользователь может изменить название организации, нажав на кнопку «Сохранить изменения». Также пользователь может удалить участника, нажав на кнопку удаления, расположенную напротив имени пользователя.

## 5.2 Руководство пользователя веб приложения

При запуске веб приложения пользователь переходит на страницу «Вход» (рис. 5.13).

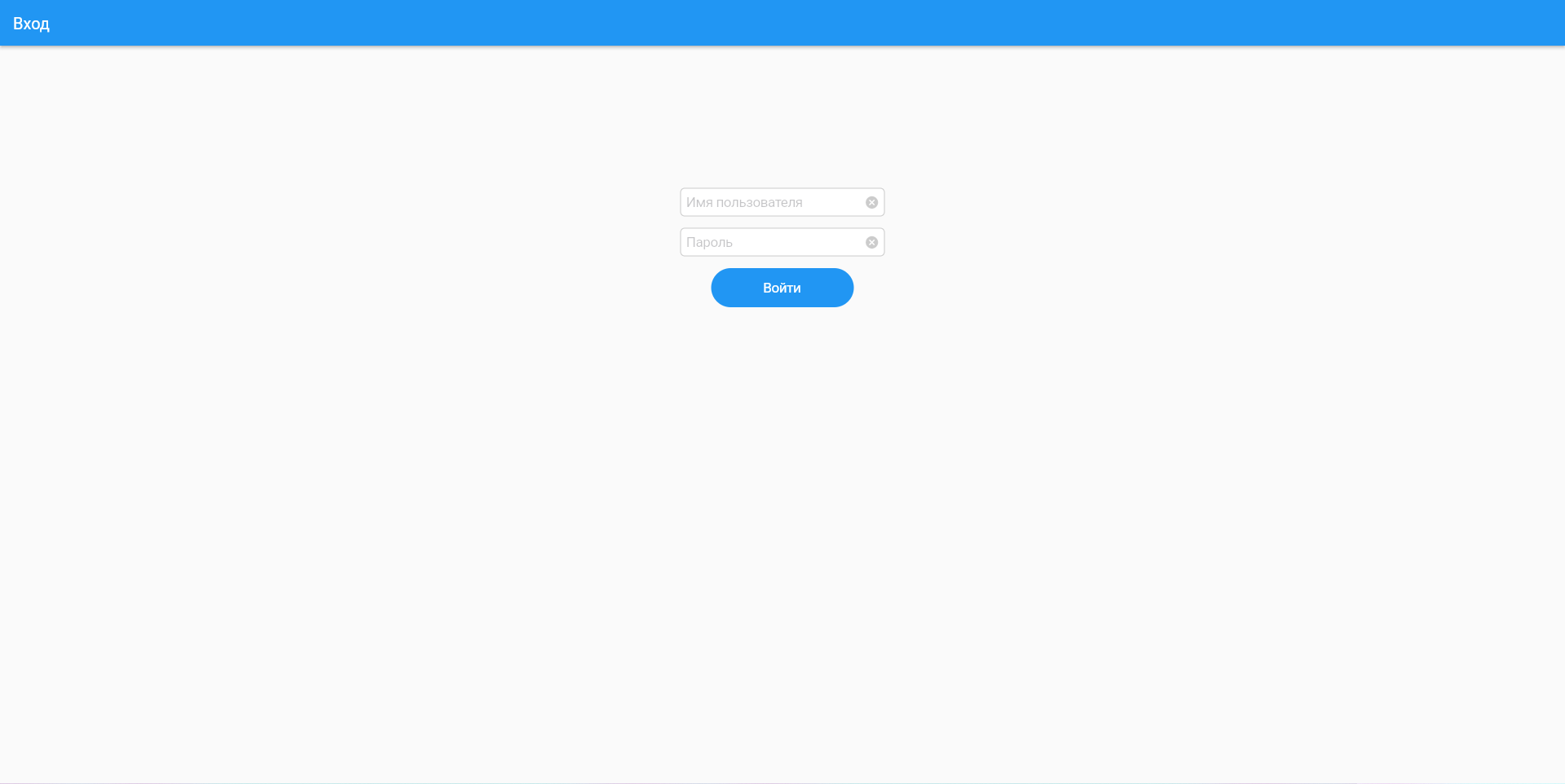


Рисунок 5.13 – страница «Вход» веб приложения

На данной странице пользователь вводит имя пользователя и пароль, после чего пользователь переходит на страницу «Все проекты» (рис. 5.14).

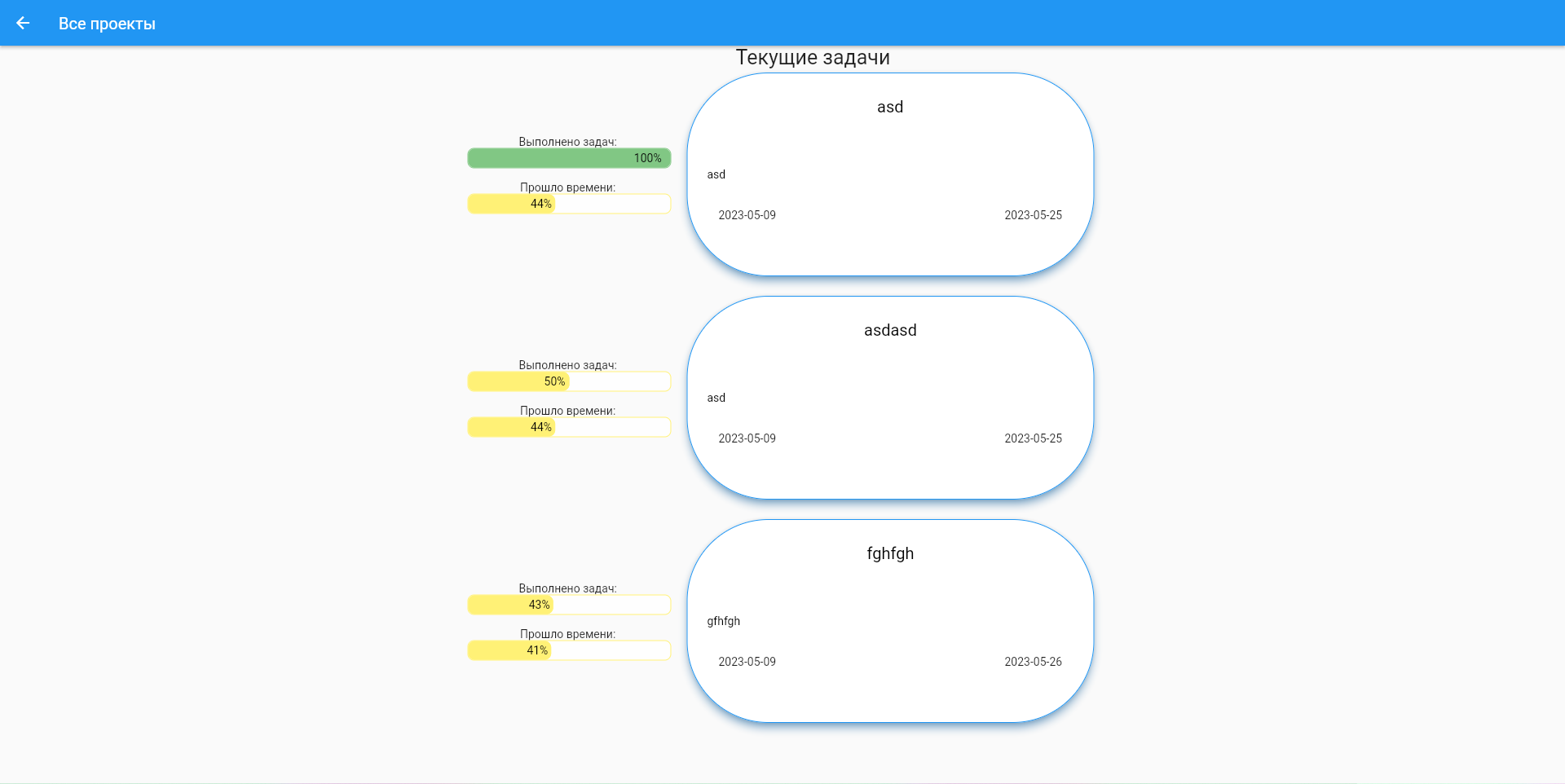


Рисунок 5.14 – страница «Все проекты» веб приложения

На данной странице пользователь может выполнить три действия: продлить дату проекта, завершить проект или удалить проект. Для выполнения данных действий пользователю необходимо совершить долгое нажатие на проект, надо которым он хочет совершить действие. После этого появится контекстное меню со списком действий (рис. 5.15).

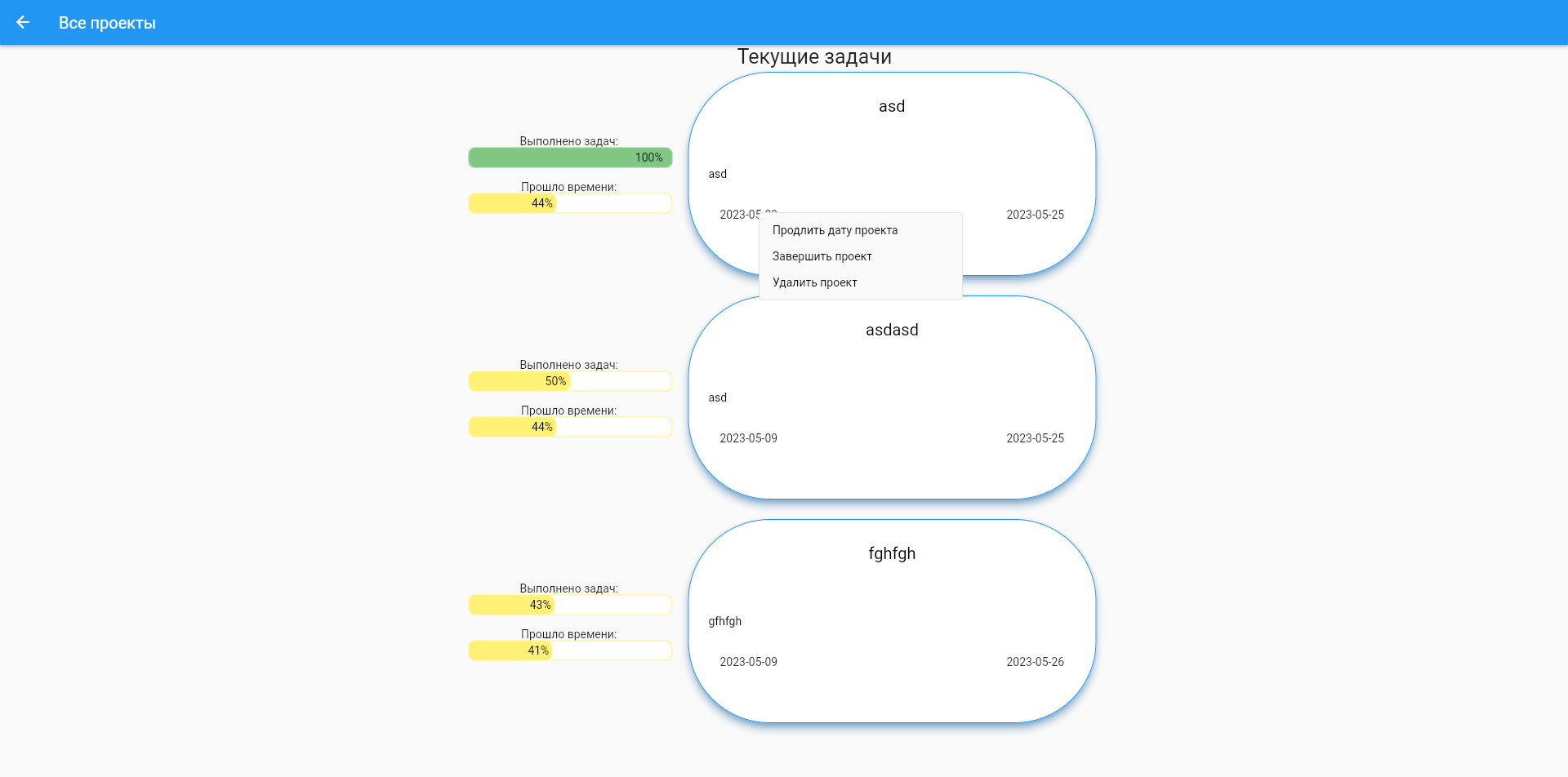


Рисунок 5.15 – контекстное меню со списком действий

При нажатии на элемент меню «Продлить дату проекта» открывается календарь для выбора новой даты завершения проекта (рис.5.16).

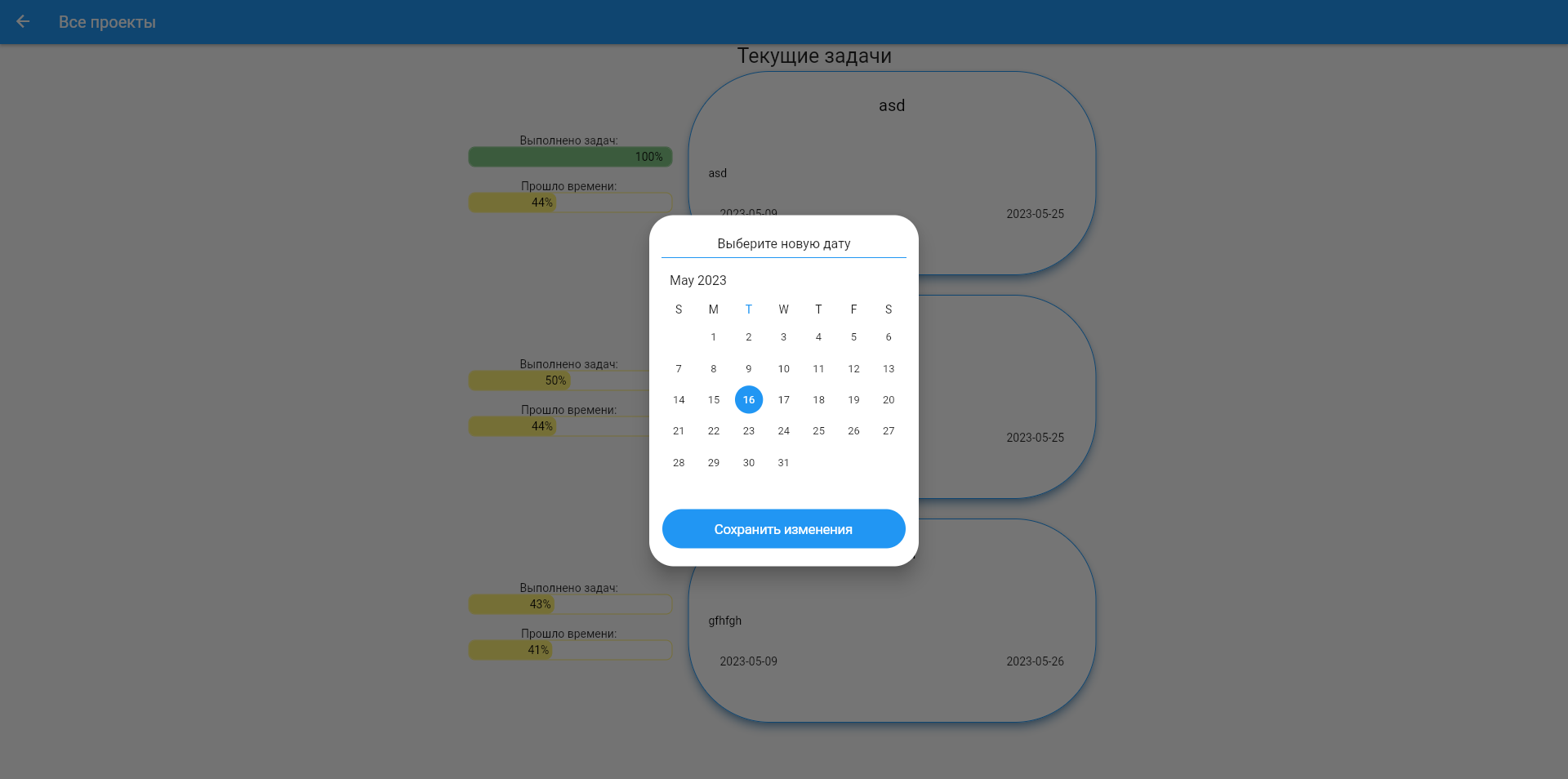


Рисунок 5.16 – календарь для выбора новой даты завершения проекта

При нажатии на элемент меню «Удалить проект» будет выполнен запрос к серверу и проект будет удален. После нажатия на кнопку «Завершить проект», будет проведена проверка, все ли задачи проекта выполнены, и в случае, если все задачи проекта выполнены значение поля isDone данного проекта будет установлено в 1 и проект будет перемещен в раздел «Выполненные задачи» (рис. 5.17).

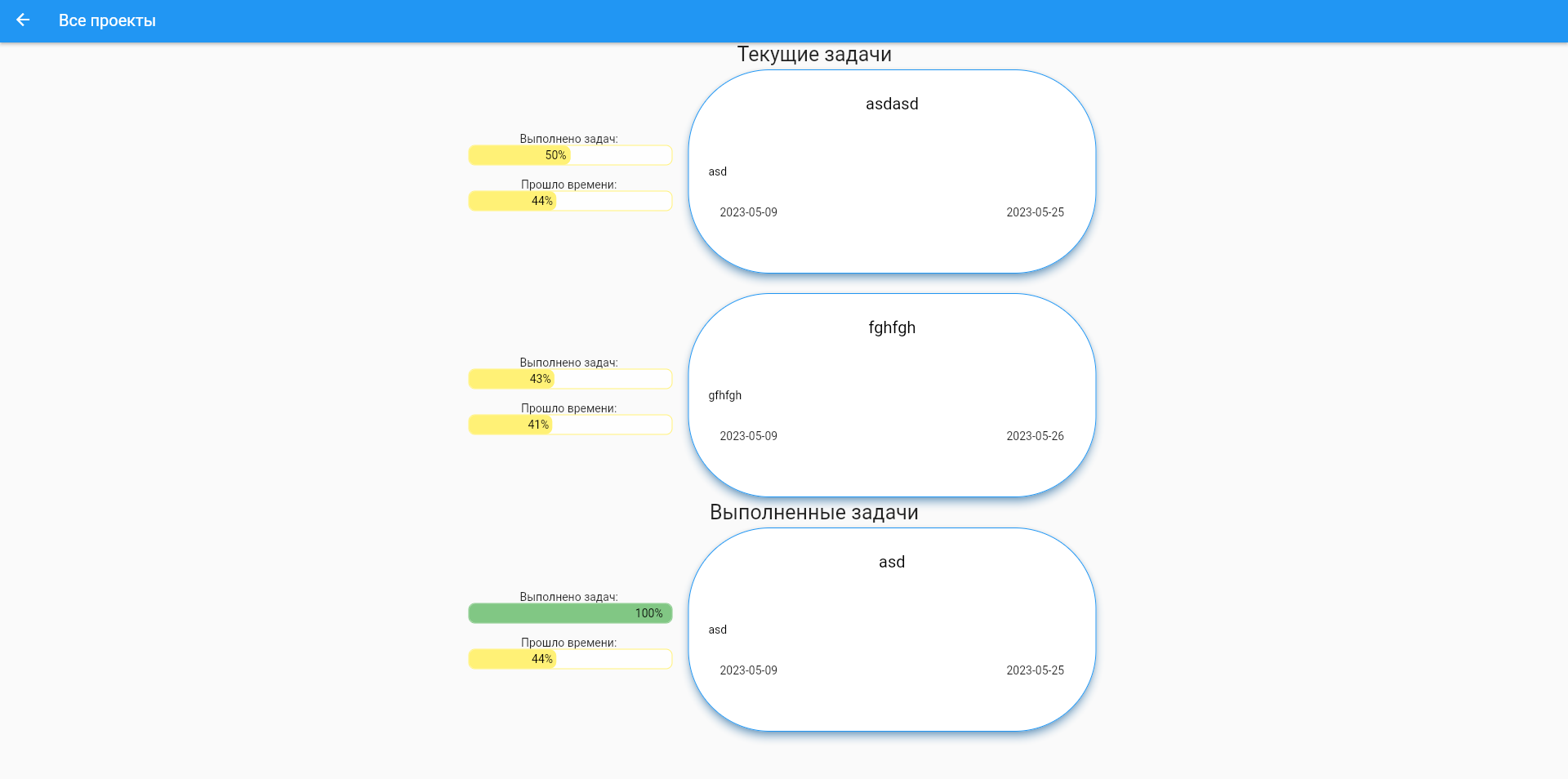


Рисунок 5.17 – раздел выполненные задачи

При нажатии на проект будет открыта страница выбранного проекта (рис. 5.18).

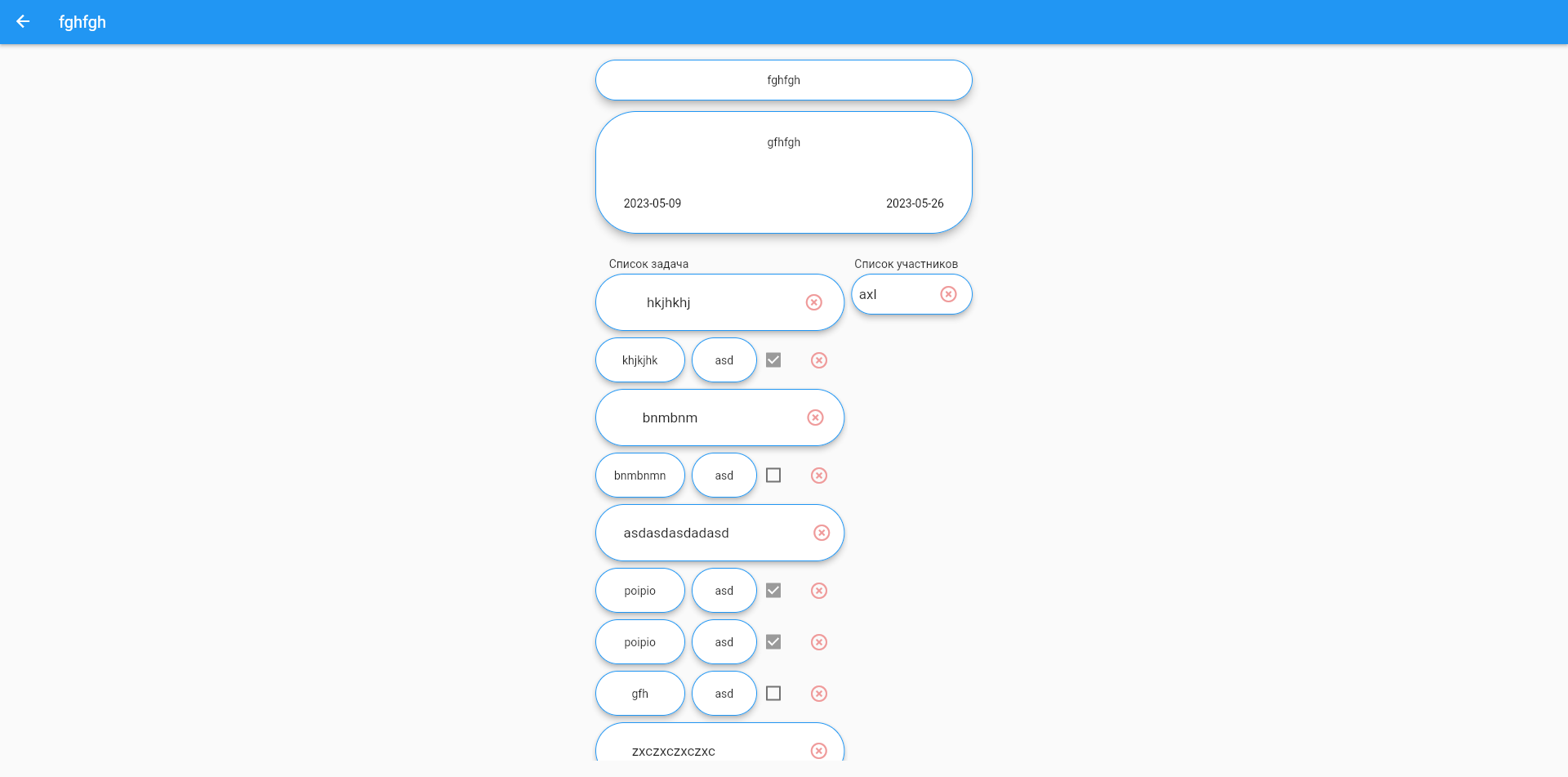


Рисунок 5.18 – страница проекта

На данной странице пользователь может просмотреть задачи и подзадачи проекта, а также удалить задачу, подзадачу или исключить пользователя из проекта. Для совершения данных действий необходимо нажать на соответствующую кнопку, расположенную напротив названия задачи или имени пользователя.

# **ЗАКЛЮЧЕНИЕ**

Целью курсового проекта была разработка мобильного приложения, предоставляющего следующие задачи:

* регистрация/авторизация пользователей;
* Создание проекта;
* Создание подзадач проекта;
* Добавление участников в проект;
* Изменение прогресса выполнения проекта;
* Удаление участников из проекта.

Также необходимо было разработать веб приложение, сервер, локальную и глобальную базы данных.

В ходе выполнения поставленных задач были разработаны макеты мобильного и веб приложений, созданы диаграммы вариантов действий.

При проектировании баз данных были разработаны необходимые таблицы, выбраны первичные и вторичные ключи для каждой из таблиц, использована технология Soft Delete.

После завершения этапа проектирования были выбраны технические для реализации поставленных задач. Для разработки глобальной базы данных была выбрана СУБД Microsoft SQL Server. В качестве средств разработки использовались Android Studio и Visual Studio Code. Для написания реализации использовались языки программирования Dart в совокупности с фреймворком Flutter и Node.js в совокупности с фреймворком Express.

Для обеспечения безопасности личных данных пользователя используется шифрования с помощью библиотеки crypto/

Мобильное приложение, кроме вышеперечисленных функций, также обладает возможностью синхронизации, которая обеспечивает возможность использования приложения без доступа к сети интернет.

В результате выполнения задач было разработано мобильное приложение, обеспечивающее следующие функции:

* регистрация/авторизация пользователей;
* Создание проекта;
* Создание подзадач проекта;
* Добавление участников в проект;
* Изменение прогресса выполнения проекта.

Также было разработано веб приложение, обладающее следующим функционалом:

* Просмотр статистики проекта;
* Завершение проекта;
* Изменение даты завершения проекта;
* Удаление проекта;
* Удаление задачи проекта;
* Удаление подзадачи проекта;
* Исключение участника из проекта.

# **Список использованных источников**

1. Интеграция с базами данных [Электронный ресурс]. / https://tediousjs.github.io/tedious/index.html – Режим доступа: https://tediousjs.github.io/tedious/api-connection.html
2. HTTP и HTTPS, сетевые запросы в Dart и Flutter [Электронный ресурс] / flutter.su – Режим доступа: https://flutter.su/tutorial/7-HTTP-network-requests
3. crypto. Flutter Package [Электронный ресурс] / pub.dev/packages – Режим доступа: https://pub.dev/packages/crypto
4. Persist data with SQLite [Электронный ресурс] / docs.flutter.dev – Режим доступа: https://docs.flutter.dev/cookbook/persistence/sqlite

# **ПРИЛОЖЕНИЕ А**

CREATE TABLE AppUser(

[id] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,

[username] [nvarchar](50) NULL UNIQUE,

[password] [nvarchar](50) NULL,

) ;

CREATE TABLE Organisation(

[id] [int] IDENTITY(1,1) NOT NULL,

[name] [nvarchar](50) NULL,

[password] [nvarchar](50) NULL,

[creatorID] [int] FOREIGN KEY REFERENCES AppUser(id),

);

CREATE TABLE OrganisationMember(

[userID] [int] FOREIGN KEY REFERENCES AppUser(id),

[organisationID] [int] FOREIGN KEY REFERENCES Organisation(id),

[deleted] [bit] NULL,

[id] [int] IDENTITY(1,1) PRIMAEY KEY,

);

CREATE TABLE ProjectMember(

[id] [int] IDENTITY(1,1) PRIMARY KEY,

[projectID] [int] NULL,

[organisationMemberID] [int] FOREIGN KEY REFERENCES OrganisationMember(id),

[deleted] [bit] DEFAULT(0)

);

CREATE TABLE Project(

[id] [int] IDENTITY(1,1) PRIMARY KEY,

[title] [nvarchar](255) NULL,

[decription] [nvarchar](255) NULL,

[startDate] [date] NULL,

[endDate] [date] NULL,

[isDone] [bit] NULL,

[creatorID] [int] NULL,

);

ALTER TABLE ProjectMember ADD FOREIGN KEY(projectID)

REFERENCES Project (id)

ALTER TABLE Project ADD FOREIGN KEY(creatorID)

REFERENCES ProjectMember (id)

CREATE TABLE SubTask(

[id] [int] IDENTITY(1,1) PRIMARY KEY,

[parent] [int] NULL,

[projectID] [int] FOREIGN KEY REFERENCES Project(id),

[title] [nvarchar](50) NULL,

[isDone] [bit] NULL,

[isTotallyDone] [bit] DEFAULT(0),

);

CREATE TABLE SubTaskExecutor(

[id] [int] IDENTITY(1,1) PRIMARY KEY,

[subTaskID] [int] FOREIGN KEY REFERENCES SubTask(id),

[executorID] [int] FOREIGN KEY REFERENCES ProjectMember(id),

);

# **ПРИЛОЖЕНИЕ** **Б**

create table AppUser(

id integer primary key AUTOINCREMENT,

username text,

password text,

changed integer

);

create table Organisation(

id integer primary key AUTOINCREMENT,

name text,password text,

creatorID integer,

changed integer,

foreign key(creatorID) references AppUser(id)

);

create table OrganisationMember(

id integer primary key AUTOINCREMENT,

userID integer references AppUser(id),

organisationID integer REFERENCES Organisation(id),

deleted integer,

changed integer

);

create table Project(

id INTEGER PRIMARY KEY AUTOINCREMENT,

title text,

decription text,

startDate date,

endDate date,

isDone INTEGER,

changed INTEGER

);

create table ProjectMember(

id INTEGER PRIMARY KEY AUTOINCREMENT,

projectID integer references Project(id),

deleted INTEGER default 0,

changed INTEGER

);

alter table Project add column creatorID integer references ProjectMember(id)

alter table ProjectMember add column organisationMemberID integer references OrganisationMember(id);

create table SubTask(

id integer PRIMARY KEY AUTOINCREMENT,

parent INTEGER null,

projectID INTEGER REFERENCES Project(id),

title text,

isDone integer,

isTotallyDone integer,

changed INTEGER,

weakcreated integer,

created integer

);

create table SubTaskExecutor(

id integer PRIMARY KEY AUTOINCREMENT,

subTaskID integer REFERENCES SubTask(id),

executorID integer REFERENCES ProjectMember(id),

changed INTEGER

);