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:**

Мобильное приложение для студентов конкретного факультета

Выполнил студент

(Ф.И.О.)

Руководитель проекта ассистент Нистюк О.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: ассистент Нистюк О.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2021

Реферат

Пояснительная записка курсового проекта содержит 41 страниц пояснительной записки, 10 источников литературы, 4 приложения.

Цель курсового проекта – разработка программных модулей, которые будут корректно взаимодействовать между собой и смогут обеспечить возможность просмотра статистики сдачи лабораторных работ, статистики сдачи по группе, просмотра автоматически строящегося плана сдачи лабораторных, добавление планов в календарь, редактирование своего профиля, это все касаемо студента, а преподаватель имеет возможность изменять количество сданных лабораторных работ определенного студента.

В первой главе проводится анализ предметной области и формирование функционала.

Вторая глава посвящена процессу проектирования системы и описанию технологий, которые будут использованы в проекте.

В третьей главе описаны подробности реализации.

В четвёртой главе описывается руководство пользователя.

В четвертой главе описано тестирование и проверка работоспособности получившихся приложений.

В заключении приведены результаты проделанной работы.

Содержание

[Реферат 2](#_Toc73283807)

[Введение 4](#_Toc73283808)

[1 Анализ и моделирование предметной области, формирование функциональных требований к проектируемому мобильному приложению 5](#_Toc73283809)

[1.1 Исследование предметной области, выявление основных проблем 5](#_Toc73283810)

[1.2 Разработка функциональных требований 6](#_Toc73283811)

[1.3 Технические требования 6](#_Toc73283812)

[1.4 Итоги раздела 7](#_Toc73283813)

[2 Проектирование приложения 8](#_Toc73283814)

[2.1 Архитектура 8](#_Toc73283815)

[2.2 Серверная часть приложения 8](#_Toc73283816)

[2.3 Безопасность. Аутентификация и авторизация 8](#_Toc73283817)

[2.4 Сущности в СУБД Microsoft SQL SERVER 9](#_Toc73283818)

[2.5 Сущности в СУБД SQLite 12](#_Toc73283819)

[2.6 Клиентская часть приложения 13](#_Toc73283820)

[2.7 Выводы по главе «Проектирование программного продукта» 14](#_Toc73283821)

[3 Программная реализация приложений 15](#_Toc73283822)

[3.1 Технические средства для разработки 15](#_Toc73283823)

[3.2 Разработка мобильного приложения 15](#_Toc73283824)

[3.4 Разработка сервера 16](#_Toc73283825)

[3.5 Вывод по главе «Программная реализация приложений» 17](#_Toc73283826)

[4. Руководство пользователя 18](#_Toc73283827)

[4.1 Руководство по мобильному приложению. 18](#_Toc73283828)

[4.2 Руководство по веб-приложению 26](#_Toc73283829)

[5. Тестирование разработанного продукта 28](#_Toc73283830)

[5.1 Негативное тестирование веб приложения 28](#_Toc73283831)

[5.1 Негативное тестирование мобильного приложения 28](#_Toc73283832)

[Заключение 31](#_Toc73283833)

[Список литературы 32](#_Toc73283834)

[Приложение A 33](#_Toc73283835)

[Приложение Б 38](#_Toc73283836)

[Приложение В 39](#_Toc73283837)

[ПРИЛОЖЕНИЕ Г 41](#_Toc73283838)

**Введение**

Цель курсового проекта: разработка программного средства «FIT.Tracker», предоставляющего возможность пользователю-студенту контролировать сдачу лабораторных работ и строить планы, а пользователю-преподавателю внасить изменения о сданных лабораторных работах.

Задачи курсового проекта:

* Проектирование программного средства;
* Реализация программного средства;
* Тестирование разработанного продукта;
* Разработка и написание руководства для пользователя.

Проектирование программного средства позволит нам построить картину как должно функционировать наше программное средство в соответствии с техническими требованиями. На этом этапе мы должны построить базу данных, структуру веб-сервера, мобильного приложения и все возможные сценарии использования программного средства. Определиться с всеми технологиями, фреймворками и библиотеками, которые мы собираемся использовать при реализации нашего приложения. Убедимся, что все технологии будут исправно функционировать между собой, не создавая конфликтных ситуаций.

При реализации приложения необходимо будет придерживаться плана, намеченного на предыдущем этапе с допущением минимальных отклонений. На этом этапе производится написание всех классов, методов и процедур. По его окончании приложение должно быть полностью готово для прохождения тестов.

После этого этапа наступает этап тестирования. Тестирование может производиться выборочно, а может с полным охватом, в зависимости от того, каким количеством времени располагает тестировщик. На этом этапе отлавливаются и исправляются баги.

После прохождения всех перечисленных этапов, должно получиться полностью рабочее и отлаженное приложение.

Руководство для пользователя разрабатывается с целью ознакомления пользователя со всеми функциональными возможностями приложения. Описанием правильного использования приложения и наглядными примерами. Написание руководство — это не маловажный этап в нашей работе так как даже если вы уверены, что с вашим продуктом легко работать, а его интерфейс интуитивно понятен – всегда найдутся пользователи, у которых возникнут вопросы к вашему программному средству.

1 Анализ и моделирование предметной области, формирование функциональных требований к проектируемому мобильному приложению

1.1 Исследование предметной области, выявление основных проблем

Приложение для контроля сданных лабораторных работ, просмотра статистики сдачи и формирования каких-то задач в календаре позволяет лучше студенту спланировать темп сдачи, а статистика по группе подстегивает некоторых студентов, чтобы всегда быть №1 и опережать план. Данное приложение создано для того, чтобы студенты в любой момент могли оценивать свои возможности и понимать, что они либо отстают в сдаче, либо все хорошо. С другой стороны, это приложение также является непосредственной связью между преподавателем и студентом, т.к. после защиты лабораторной работы преподаватель вносит изменения в базу данных и у студента меняется статистика и т.п. Также данное приложение можно охарактеризовать как некий дневник студента, где преподаватель засчитывает или нет, сдаваемую студентом, лабораторную.

Немаловажным этапом в разработке программного продукта является аналитический обзор прототипов. В качестве прототипа для анализа можно рассмотреть приложение “Расписание для студентов” [7]. Приложение имеет следующие возможности:

* добавление расписания занятий;
* добавление оценок, посещаемости;
* просмотр списков преподавателей, предметов, аудиторий и т.п.

Интерфейс приложения представлен на рисунке 1.1.

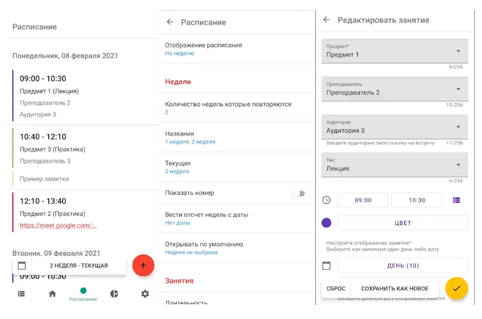


Рис. 1.1 – «Приложение Расписание для студентов»

Недостатки приложения:

* Отсутствие встроенного расписания;
* Отсутствие формирования автоматически плана сдачи лабораторных;
* Присутствие в большом количестве ненужной информации;

1.2 Разработка функциональных требований

После того как выяснены основные проблемы в области и определён фундаментальный функционал, необходимо его конкретизировать и преобразовать в функциональные требования, а также разработать дополнительные функции для того, чтобы улучшить пользовательский опыт.

В итоге был сформирован следующий функционал:

* регистрация пользователя
* авторизация
* возможность редактирования данных пользователя
* просмотр расписания лабораторных занятий
* просмотр более детализированной информации о лабораторной с статусом сдачи
* просмотр авто-генерирующегося, на основе данных семестра и дня сдачи, плана сдачи лабораторных
* возможность добавления ланов в календарь
* возможность просмотра статистики сдачи лабораторных по подгруппе
* возможность просмотра всех групп и всех студентов, относящихся к этим группам, на стороне преподавателя
* возможность изменения количества сданных лабораторных работ, на стороне преподавателя

Помимо самого приложения для смартфона, необходимо сделать сервер, который будет ответственен за реализацию всей логики синхронизации, поддержку аутентификации и авторизации, взаимодействию с базой данных.

1.3 Технические требования

Многие функции мобильного приложения не требуют взаимодействия с сервером, который в свою очередь взаимодействует с базой данных, следовательно, для работы приложения не требуется стабильное подключение к интернету. Операционная система Android имеет множество версий, чем ниже версия системы, под которую будет разрабатываться приложение, тем больше устройств смогут её запустить, но тем меньше новых функций будет доступно при разработке. Таким образом, оптимальным выбором оказалось версия под номером 24.

Серверная часть приложения будет разработана с использованием программной платформы, основанной на C#, а язык, использующийся при разработке android приложения – Java 11. Вместе с сервером будет работать СУБД Microsoft SQL Server.

Как мобильное приложение, так и сервер не требуют много ресурсов для своей работы, поэтому ставить вопрос о комплектующих, необходимых для работы приложения и сервера, не имеет смысла.

1.4 Итоги раздела

По итогам данного раздела была проанализирована предметная область, в ней выявлены ключевые проблемы, которые необходимо устранить, и на основе этого был сформирован функционал программы, а также определены базовые технологии для его реализации.

2 Проектирование приложения

Проектирование приложения – самая важная часть разработки, ведь именно на этапе проектирования выявляются как мелкие огрехи, так и крупные недоработки, исправление которых на этапе его развёртывания будет крайне затруднительно и накладно. Важность проектирования в данной курсовой работе очень высока ещё и потому, что разрабатываться будут три компонента: мобильное приложение, серверная часть, а также веб-приложение. Необходимо продумать то, как они будут без помех взаимодействовать друг с другом.

2.1 Архитектура

Для разработки подобного типа проекта было решено использовать стандартную трёхуровневую архитектуру [3], так как такая архитектура позволяет добиться низкой связности уровней, что, в свою очередь, обеспечивает для приложения надёжность, простоту долгосрочной поддержки, а также оперативное устранение неисправностей при необходимости.

Клиент представляет из себя либо мобильное, либо веб-приложение с пользовательским интерфейсом, которое будет совершать REST-запросы на сервер и обрабатывать их в читаемый для пользователя вид.

Сервер будет содержать в себе бизнес-логику по поддержке аутентификации и авторизации, а также будет ответственным за всё взаимодействие с базой данных.

2.2 Серверная часть приложения

При проектировании серверной части приложения необходимо стараться держать низкий уровень избыточности и связности между составляющими частями, а также усиленно следить за корректностью кода, поскольку от этого будет зависеть работа всех подключенных клиентов.

Для того, чтобы обмениваться данными между клиентами и сервисами был выбран архитектурный стиль REST. По своему определению он широко использует возможности протокола HTTP и поэтому не имеет состояния (stateless). Это современный подход, использующийся в большинстве веб-сервисов и поэтому имеющий множество примеров реализаций различных функций на его основе, таких как аутентификация, выполнение CRUD-операций и т.д. В REST-приложениях в подавляющем большинстве случаев используется JSON для передачи данных, его также было решено взять как формат данных при передаче между клиентами и сервером.

2.3 Безопасность. Аутентификация и авторизация

Важнейшим аспектом проектирования серверной части приложения является то, как пользователи будут входить в систему и как система будет понимать, что очередной присланный запрос прислан уже находящимся в системе пользователем.

Стандартными подходами при реализации этих функций является аутентификация с помощью логина и пароля, при этом используя собственную БД в качестве их места хранения.

После того, как пользователь совершит вход в приложение, необходимо поддерживать его статус и позволять выполнять запросы к API, в то время как запросы не аутентифицированных пользователей – отклонять. Как было описано ранее, серверная часть работает без сохранения состояния по протоколу HTTP.

2.4 Сущности в СУБД Microsoft SQL SERVER

Изучение предметной области и составленный впоследствии функционал показали, что можно выделить 15 таблиц, которые позволяют полностью охватить обозначенный функционал. Из них:

* Семестры (Semestr) – таблица, предназначенная для хранения основных данных о семестрах,;

Столбцы данной таблицы:

IDSem – целочисленный тип данных, является первичным ключом, значения которого варьируются от 0 до 3, не включая;

MonthDayStart – строковый тип данных, максимальная длина 5;

MonthDayEnd – строковый тип данных, максимальная длина 5;

* Курсы (Cource) – таблица, предназначенная для хранения курсовстудентов.

Столбцы данной таблицы:

IDcource – уникальный идентификатор, является первичным ключом;

NumberCource – целочисленный тип, варьирующийся от 0 до 5;

* Специальность (Speciality) – таблица, предназначенная для хранения специальности студентов;

Столбцы данной таблицы:

NameSpeciality – строковый тип данных, является первичным ключом, максимальная длина 100

* Подгруппы (SubGroup) – таблица, предназначенная для хранения подгрупп студентов.

Столбцы данной таблицы:

IDSubGroup – уникальный идентификатор, является первичным ключом, варьирующийся от 0 до 3;

* Группы (Goups) – таблица, предназначенная для хранения группы студентов;

Столбцы данной таблицы:

IDGroup – целочисленный тип данных, является первичным ключом

NumberGroup – целочисленный тип данных, варьирующтйся от 0 до 11

IDSpeciality – строковый тип данных, максимальная длина 100

IDSubGroup – целочисленный тип данных

IDCource – целочисленный тип данных

* Список Студентов (ListStudents) – таблица, предназначенная для хранения списка студентов.

Столбцы данной таблицы:

IDStudent – уникальный идентификатор, является первичным ключом;

Name – строковый тип, максимальная длина 50;

Surname – строковый тип, максимальная длина 50;

IDGroup – целочисленный тип;

* Специальность (Student) – таблица, предназначенная для хранения специальности студентов;

Столбцы данной таблицы:

Login – строковый тип данных, является первичным ключом, максимальная длина 50

Password – целочисленный тип данных

IDStudent – целочисленный тип данных

Email – строковый тип данных, является первичным ключом, максимальная длина 100

AboutMe – строковый тип данных, является первичным ключом, максимальная длина 250

* Список лабораторный (ListLab) – таблица, предназначенная для хранения списка лабораторных.

Столбцы данной таблицы:

IDLab – уникальный идентификатор, является первичным ключом;

NameLab – строковый тип, максимальной длинны 100;

Quantity – целочисленный тип;

IDCource – целочисленный тип;

IDSpeciality – строковый тип, максимальной длинны 100;

IDSem – целочисленный тип;

* План сдачи (PlansPass) – таблица, предназначенная для хранения специальности студентов;

Столбцы данной таблицы:

IDPlan – целочисленный тип данных, является первичным ключом

IDGroup – целочисленный тип данных

IDLab – целочисленный тип данных

Date – строковый тип данных, максимальная длина 10

Quantity – целочисленный тип данных

* Списки преподавателей (ListTeachers) – таблица предназначена для хранения списка ппреподователей;

Столбцы данной таблицы:

IDTeacher - целочисленный тип данных, является первичным ключом;

Name - строковый тип, максимальный размер 50;

Surname – строковый тип, максимальный размер 50;

* Преподаватели (Teacher) – таблица предназначена для хранения залогиненых преподавателей;

Столбцы данной таблицы:

Login – строковый тип данных, является первичным ключом;

Password - целочисленный тип;

IDTeacher – целочисленный тип;

Email – строковый тип данных;

AboutMe - строковый тип данных;

* Список лабораторных преподавателя (ListLabTeacher) – таблица предназначена для хранения списка лабораторных, которые ведет определенный преподаватель;

Столбцы данной таблицы:

IDListLabTeacher - целочисленный тип данных, является первичным ключом;

IDTeacher - целочисленный тип;

IDGroup – целочисленный тип;

IDLab – целочисленный тип данных;

WeekName - строковый тип данных;

* Сданные студентом (StudentPass) – таблица предназначена для хранения сданных студентом лабораторных работ;

Столбцы данной таблицы:

IDStudentPass - целочисленный тип данных, является первичным ключом;

IDStudent - целочисленный тип;

IDTeacher – целочисленный тип;

IDLab – целочисленный тип данных;

PassedQuantity - целочисленный тип данных;

* Администратор (Admin) – таблица предназначена для хранения логина и пароля админа

Столбцы данной таблицы:

Login - строковый тип данных, является первичным ключом;

Password - целочисленный тип;

* Дни недели (weekDay) – таблица предназначена для хранения дня недели

Столбцы данной таблицы:

IDWeek - строковый тип данных, является первичным ключом;

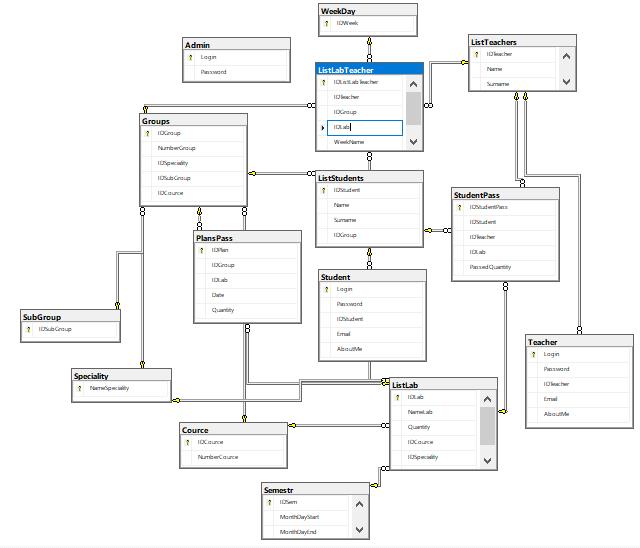


Рисунок 2.4.1 – Схема базы данных Microsoft SQL Server

2.5 Сущности в СУБД SQLite

Следующий этап разработки – проектирование локальной базы данных мобильного приложения. Все данные мобильного приложения будут заноситься в локальную базу данных SQLite. При синхронизации данные будут дублироваться на удалённую базу данных через сервер на C#. Также требуется запись в локальную базу данных изменений, сделанных пользователем через веб приложение. С учётом вышеуказанных условий была разработана следующая структура для локальной БД (в данном проекте используется SQLite. Данные таблицы БД указана на рисунке 2.3.

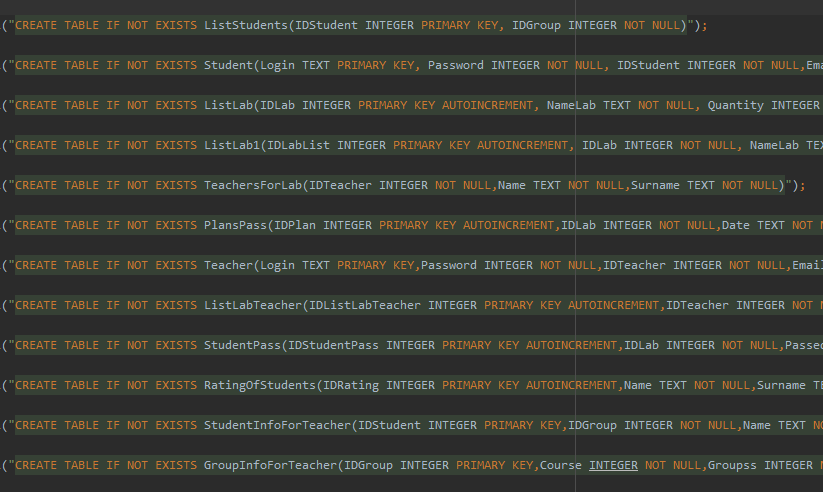


Рисунок 2.5.1 – Схема базы данных SQLite

Для работы механизма синхронизации в локальной базе данных SQLite были использованы курсоры.

2.6 Клиентская часть приложения

Клиентская часть приложения не менее важна, чем серверная. Это то, с чем будет контактировать пользователь, то, от чего в первую очередь зависит будут ли впечатления от использования положительными или отрицательными. Здесь стоит понимать, исходя из специфики проекта, в первую очередь разрабатывается мобильное приложение и самые строгие требования со стороны пользователя будут предъявляться именно к нему.

Точно так же, как и серверная часть приложения, как и практически любой другой проект, который имеет риск сильно разрастись в будущем, лучше всего разбить логику клиентской части приложения на части таким образом, чтобы на этой основе можно было каждую часть реализовывать последовательно и независимо от других частей. Приняв во внимание все требования к функционалу, а также лучшие практики и паттерны, использующиеся при разработке Android-приложений, логику клиентской части приложения можно разделить на категории:

* классы и разметка фрагментов;
* классы и разметка диалогов;
* адаптеры списков;
* классы, отправляющие запросы к API;
* классы, взаимодействующие с локальной БД;
* классы-репозитории;
* прочие дополнительные и вспомогательные классы.

В ходе проектирования была разработана UML-диаграммой, а именно диаграммой вариантов использования. Она отображает функциональность программного продукта с точки зрения получения значимого результата для пользователя. (рис. 2.6.1)

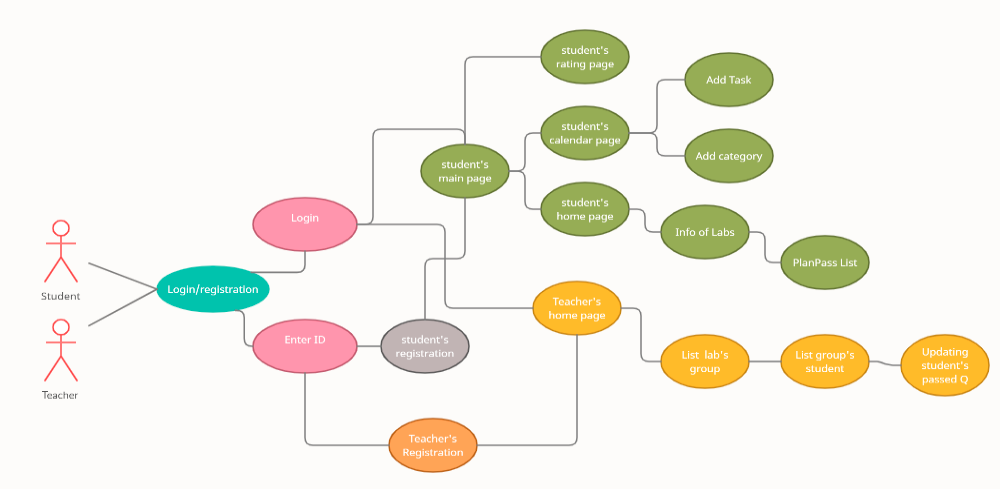


Рисунок 2.6.1 – Диаграмма вариантов использования

2.7 Выводы по главе «Проектирование программного продукта»

В данной главе было рассмотрено проектирование каждого отдельного компонента программного продукта.

По части мобильного приложения был определен основной функционал приложения, также были разработаны диаграмма вариантов использования приложения.

Была также спроектирована база данных, где были определены необходимые таблицы, охарактеризованы поля каждой из них, определены типы данных. Также были установлены связи между таблицами, заключающиеся в создании первичных и внешних ключей.

При проектировании серверного приложения был определен архитектурный стиль взаимодействия между клиентом и сервером, формат передачи данных, обозначили HTTP методы, которые будут использованы в работе с серверной частью.

3 Программная реализация приложений

3.1 Технические средства для разработки

Для разработки приложений курсового проекта были использованы следующие технические средства:

SQLite – компактная встраиваемая реляционная база данных.

Microsoft SQL Server 2018 – система управления реляционными БД.

MSSQL Server Managment Studio 2018 – СУБД MSSQL Server.

SQL – декларативный язык программирования, применяемый для создания, модификации, и управления данными в РБД.

Android Studio – интегрированная среда разработки для работы с платформой Android.

Java – строго типизированный объектно-ориентированный язык программирования.

Retrofit – REST клиент для android и Java от компании Square. Он может относительно легко получать и разбирать JSON (или другие структуированные данные) через вебсервисы, использующие REST.

Postman – набор инструментов тестирования API.

Visual Studio Code – редактор исходного кода, разработанный Microsoft для Windows

3.2 Разработка мобильного приложения

Мобильное приложение было реализовано в среде разработки – Android Studio 4.1. Для написания приложения был использован язык программирования Java. Графическая структура проекта представлена на рисунке 3.3.1.

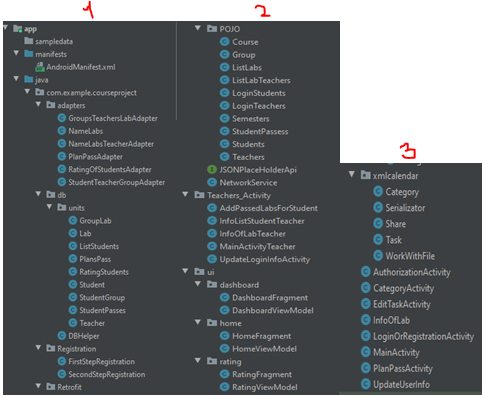


Рисунок 3.3.1 – Структура проекта мобильного приложения

Для обеспечения REST-запросов к серверу используется популярный Framework «Retrofit-2». HTTP является stateless протоколом (не хранит состояние.

Почти весь визуальный интерфейс выполнен при помощи фрагментов (fragments), либо активностей (activity).

Классы для работы с базой данных находятся в пакете db. Класс DBheper обеспечивает создание базы данны при первом запуске приложения. Для максимального быстродействия было принято решения отказаться от ORM библиотека Room и реализовать взаимодействие с локальной БД нативно.

Алгоритм синхронизации. Все изменения пользователя в локальной базе данных логируются. Когда пользователь начинает синхронизацию происходит проверка Интернет-соединения, если оно активно, все данные отправляются на сервер.

3.4 Разработка сервера

Сервер представляет из себя реализацию RESTfull API сервера. Реализован в среде разработки Visual Studio 2019, на языке C#.

Главная задача сервера – обрабатывать запросы, формировать ответы инкапсулирую логику работы с удалённой базой данных MS SQL SERVER.

Также были реализованы обработчики маршрутов на основные HTTP методы (GET, PUT, POST, DELETE) для создания RESTful сервиса.

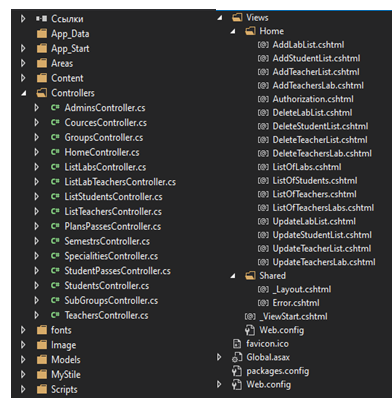


Рисунок 3.4.1 – Структура проекта сервера

3.5 Вывод по главе «Программная реализация приложений»

В данной главе было рассмотрена разработка каждого отдельного компонента приложения. На первоначальном этапе были рассмотрены технологии, которые применены для разработки курсового проекта. Вторым этапом была разработка структуры базы данных на стороне сервера и клиента. Третьим этапом стала разработка мобильного приложения, где была представлена графическая структура приложения, рассмотрены основные классы и их методы. Четвертый этап – это разработка серверного приложения, где была представлена графическая структура приложения, рассмотрены основные класс и некоторые методы.

4. Руководство пользователя

4.1 Руководство по мобильному приложению.

Для того, чтобы установить приложение на телефон, необходимо иметь версию Android 7.0 и выше (версию API 24 и более). После установки приложения на экране появится иконка и подпись «FIT LabTracker»

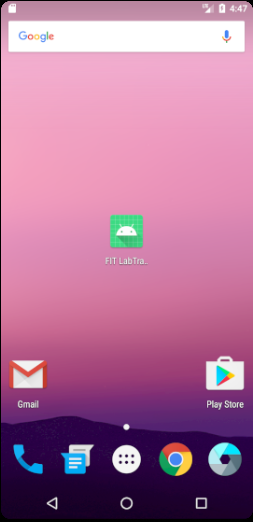


Рисунок 4.1.1 – Иконка приложения

При первом запуске приложения на экране появиться выбор в виде 2х кнопок, хотим мы зарегистрироваться или же авторизоваться.

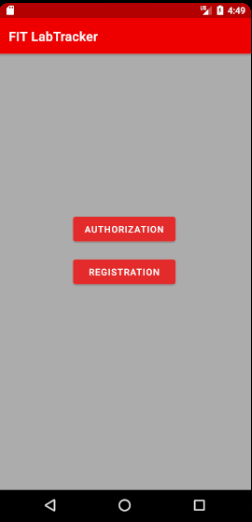


Рисунок 4.1.2 – Первое открытие приложения

Далее, если выбрать регистрацию, то мы попадем на страницу с вводом нашего ID, т.к. данное приложение является специализированным, только для определенного факультета, то зарегистрироваться могут только люди, являющиеся студентами или же преподавателями данного ВУЗа. Таким образом регистрация происходит в 2 этапа и на втором приложение само определяет твое имя и фамилию по ID, данные этапы показаны на рисунке 4.1.3

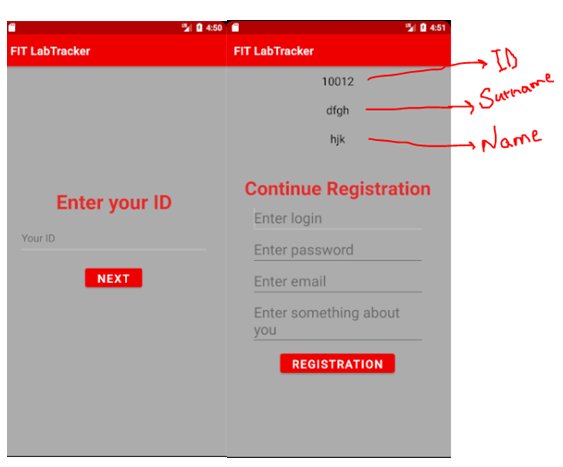


Рисунок 4.1.3 – Регистрация

При выборе авторизации появляется экран с вводом логина и пароля.

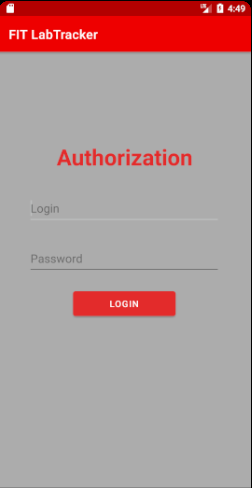


Рисунок 4.1.4 – Авторизация

Далее мобильное приложение разделяется в зависимости от того, кто в него вошел. В случае если мы авторизировались или зарегистрироввались в качестве студента, то у нас появляется экран с информацией о пользователе (курс, специальность, группа, подгруппа) и листингом выводится расписание занятий отфильтрованное по дню сдачи.

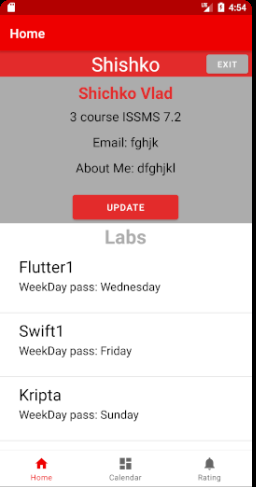


Рисунок 4.1.5 – Главная страница студента

На данной странице рисутствует кнопка “Exit”, которая при нажатии отправит нас на самую первую страницу выбора регистрации/авторизации. А также кнопка “Update”, которая предназначена для редактирования пользовательской информации, данная страница предсталенна на рисунке 4.1.6

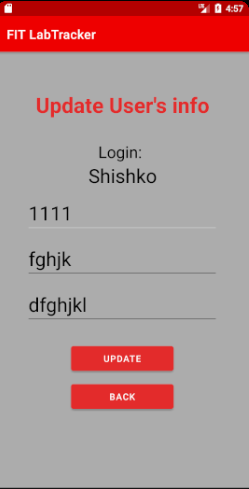


Рисунок 4.1.6 – Редактирование пользователя

Также на главной странице студента можно нажать на один из пунктов списка для просмотра более подробной информации о лабораторной, а также для просмотра успеваемости по ней

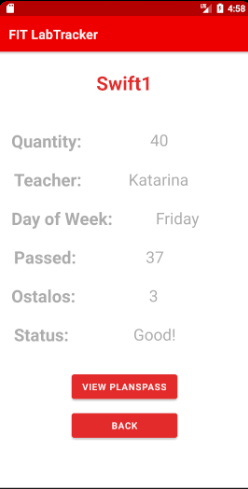


Рисунок 4.1.7 – Информация о лабораторной

На данной странице можно заметить 2 кнопки: “Back” и “View PlanPass”. При нажатии на первую из наи вы вернетесь на предыдущий этап(главная страница студента), при нажатии на 2ю вым предоставится список с автоматически остроенным планом сдачи данной лабораторной (используя эти данные оределяется то, успеваешь ты сдавать лабораторные или нет). Страница с плном представлена на рисунке 4.1.8

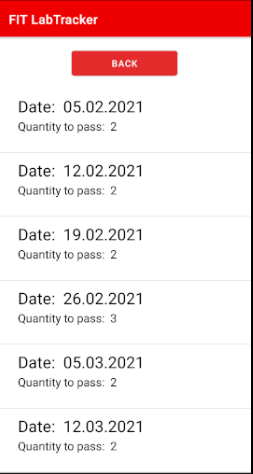


Рисунок 4.1.8 – Авто-сгенерированный план сдачи

Далее, если ввернемся на главную страницу студента, то внизу экрана можно увидеть меню, если нажать на 2й пункт – “Calrndar”, то вы попадете на страницу, где можно дабавлять какие-то свои заметки и планы по категориям, составленными вами. Данная страница представлена на рисунке 4.1.9

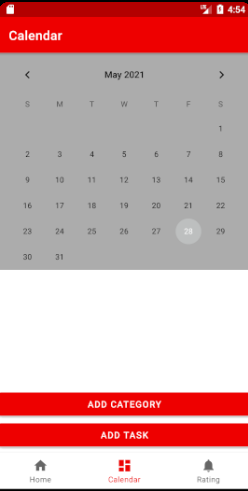


Рисунок 4.1.9 – Календарь

Добавление категории и самой задачи показаны на рисунке 4.1.10

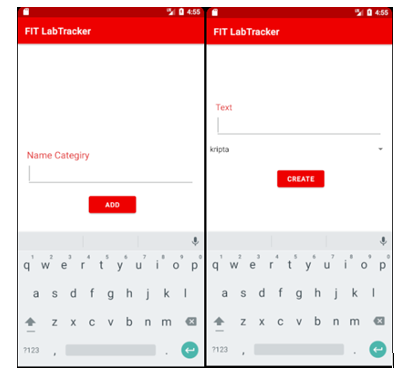


Рисунок 4.1.10 – Добавление задач/планов

В результате добавления получим следующий элемент списка:

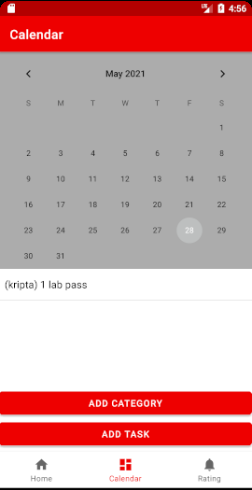


Рисунок 4.1.11 – Результат добавления

При нажатии на одну из задач/планов, можно либо удалить, либо изменить его, как показано на рисунке 4.1.12

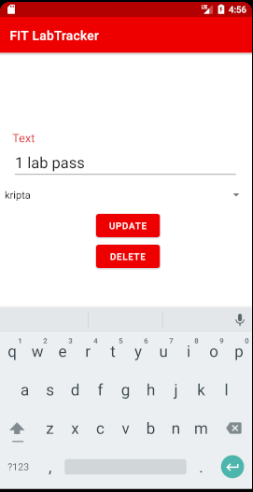


Рисунок 4.1.12 – Редактирование планов

Также если вернемся с страницы изменения планов назад, то можно увидеть 3й пункт меню – “Rating”, на данной странице отображается рейтинг студента по подгруппе

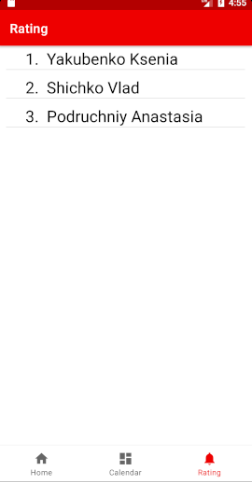


Рисунок 4.1.13 – Рейтинг студентов о подгрупппе

На этом функциональность нашего студента заканчивается. Вернемся на несколько шагов назад и посмотрим, что может делать преподаватель:

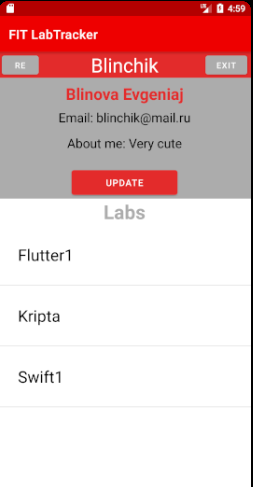


Рисунок 4.1.14 – Главная страница преподавателя

На данном рисунке можно заметить, что главные страницы студента и преподавателя относительно одинаковые. Тут также выводится краткая информация пользователя, имеется кнопка “Back” с такой же функциональностью и также кнопка “Update”. И при нажатии на один из пунктов нашего списка мы попадем на расписание данных занятий с номерами групп в которых данный преподаватель их роводит. Пример такой страницы оказан на рисунке 4.1.

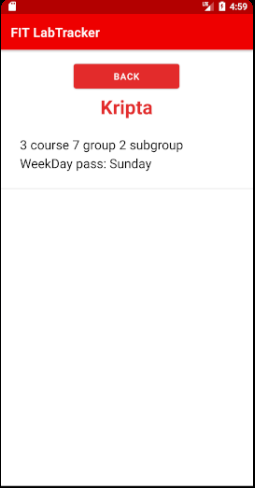


Рисунок 4.1.15 – Список групп, у которых ведет лабораторную

При нажатии на данный пункт, отобразятся все студенты данной группы, при нажатии на каждого из который предоставится возможность изменить количество сданных лабораторных.

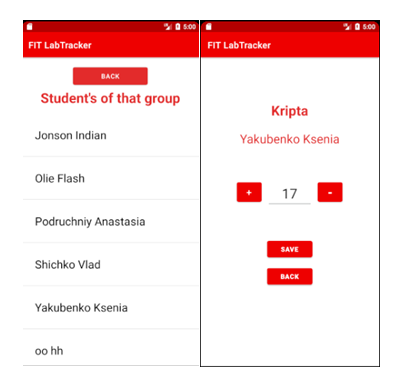


Рисунок 4.1.16 – Редактирование количества сданных лабораторных

4.2 Руководство по веб-приложению

На рисунке 4.2.1 представлено окно авторизации web-приложения. Мое web-приложение рассчитано только на администратора и он может только авторизироваться. Страница авторизации представлена на рисунке 4.2.2

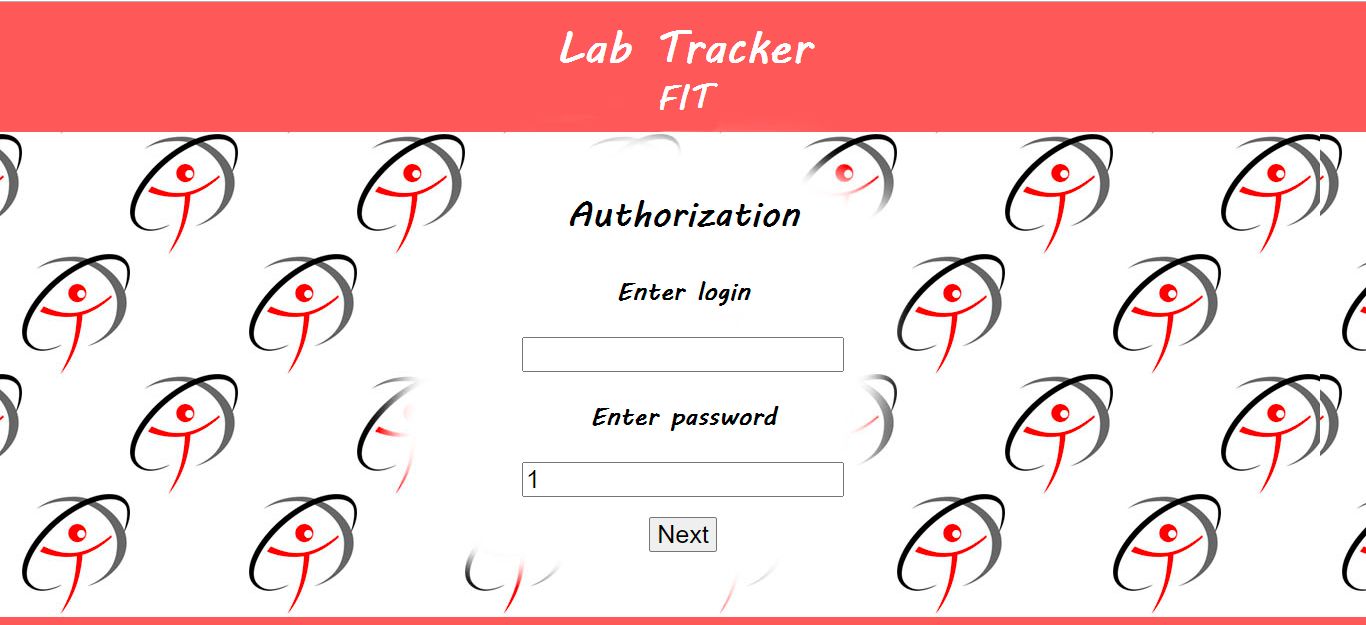


Рисунок 4.2.1 – Страница авторизации

После ввода в аккаунт админестратору предоставится возможность добавлени, редактирования и удаления 4х различных страниц: список студентов, учителей, лабораторных и списка лабораторных, которые привязаны к определенному преподавателю и группе студентов. Одна из страниц с таким списком продемонстрирована на рисунке 4.2.2

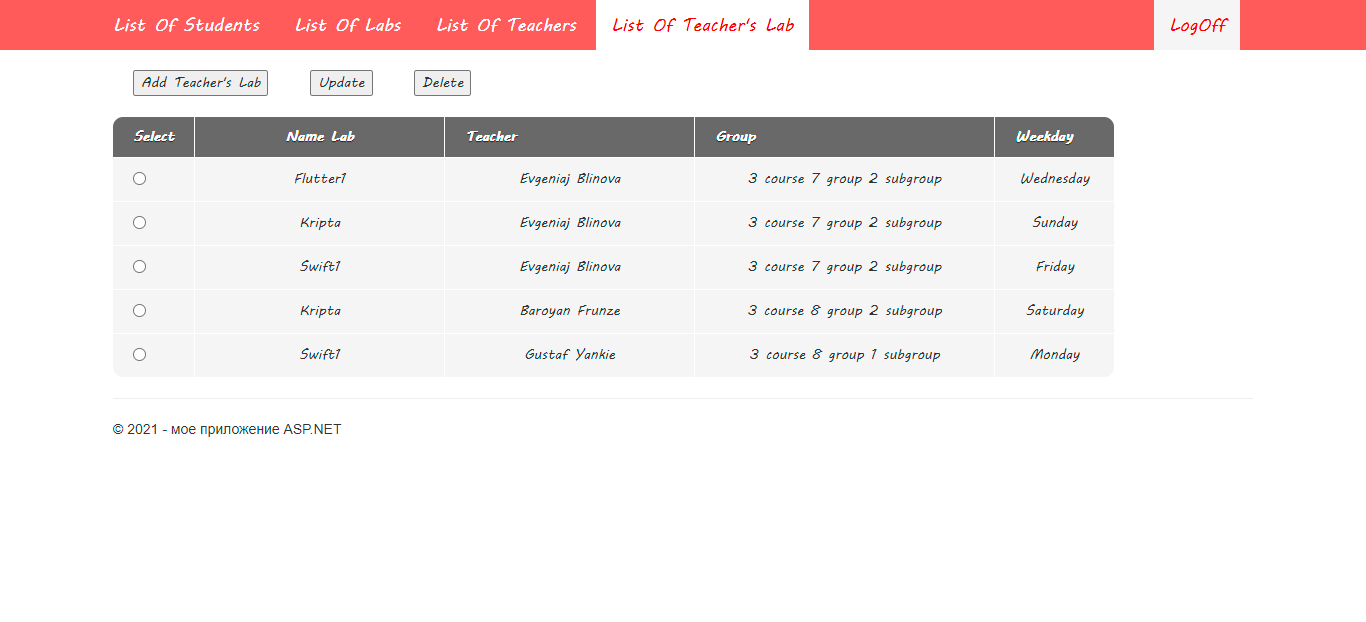


Рисунок 4.2.2 – Страница сдачи лабораторных

Как можно заметить вверху нашей страницы имеется меню, где можно ереходить между различными таблицами. А также можно заметить 3 кнопки: “Add”, “Update”, “Delete”. При нажатии на первую из них предоставляется соответственно добавление записи в таблицу, на вторую – редактирование и на 3-ю – удаление. Одна из данных страниц представленна на рисунке

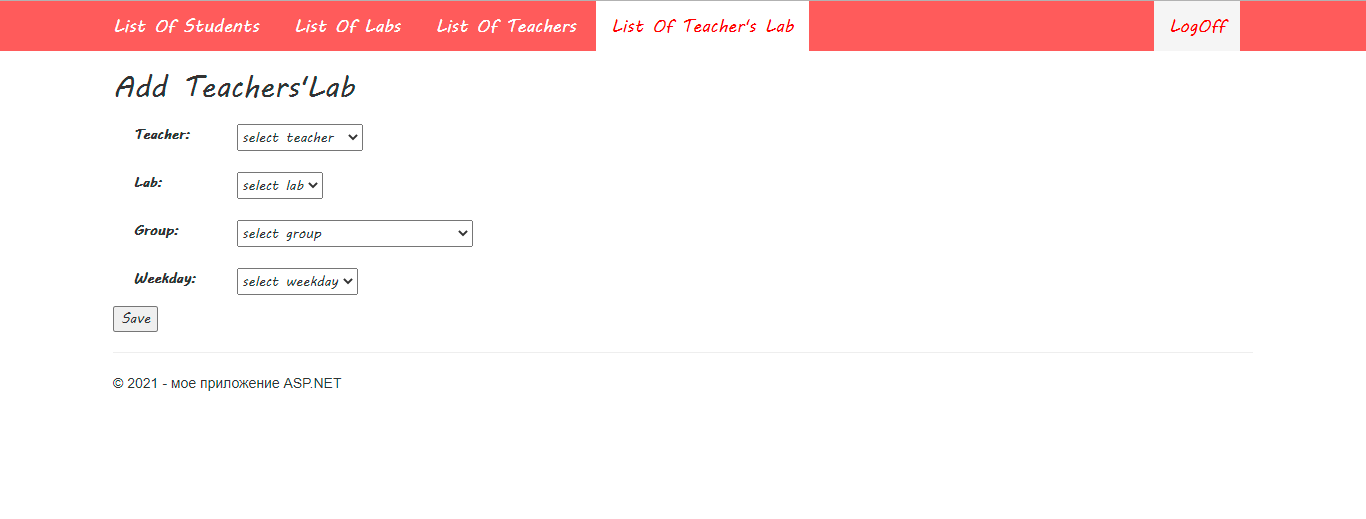


Рисунок 4.2.3 – Добавление сдачи лабораторной

5. Тестирование разработанного продукта

Процесс тестирования продукта представляет собой столь же неотъемлемую часть процесса разработки, как и проектирование. Тестирование позволяет оценить качество разрабатываемого продукта. Тестирование проекта выполнялось в течение разработки посредством таких средств, как отладка и логгирование. После введения нового либо изменения существующего компонента программы, он и зависящий от него код также подвергался тестированию. Анализ полученных результатов происходил по факту получения ошибок и проблемные места сразу исправлялись. После завершения разработки программы по нескольку раз были подвергнуты испытаниям все элементы управления в пользовательском интерфейсе. Если происходили ошибки, то их причина и местоположение в коде легко фиксировались благодаря ведению логов.

5.1 Негативное тестирование веб приложения

Негативное тестирование – это один из видов тестирования, который позволяет проверить систему на некорректное поведение. В ходе такого тестирования мы можем узнать, что система справится с непредвиденными ситуациями. Первый сценарий для тестирования: Когда мы пытаемся добавить преподавателя или студента с таким же ID, то высветится сообщение о том, чтобы мы попробовали ввести другие данные

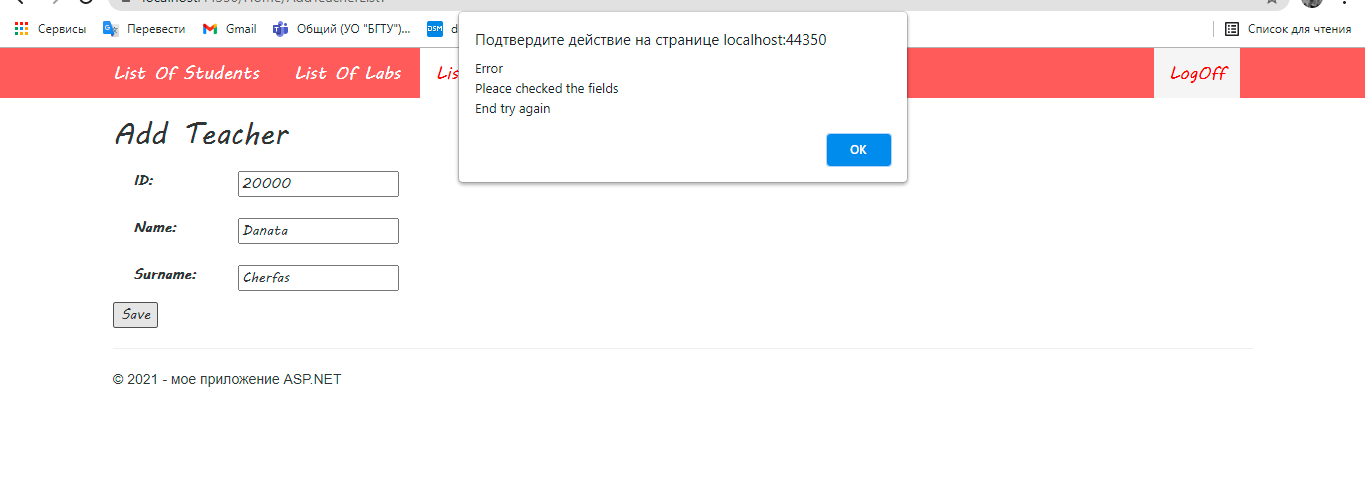


Рисунок 5.1.1 – Сообщение пользователю об ошибке

5.1 Негативное тестирование мобильного приложения

Первое что необходимо проверить, это станицы регистрации и авторизации. Для начала проверим регистрацию. Если пользователь введет не свой ID или ID, который уже зарегестрирован, то ему высветится соответствующее уведоиление, как показано на 5.2.2

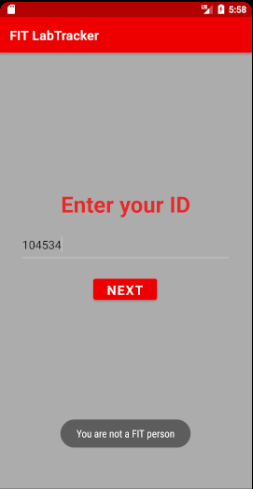
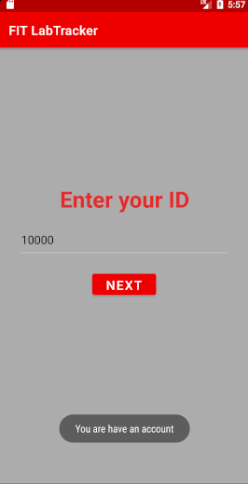


Рисунок 5.1.2 – Сообщение пользователю об ошибке

При проверки страницы автризации можно проверить случай, если пользователь забудет ввести пароль, тогда ему об этом сообщит также уведомление, которое продемонстрировано на рисунке

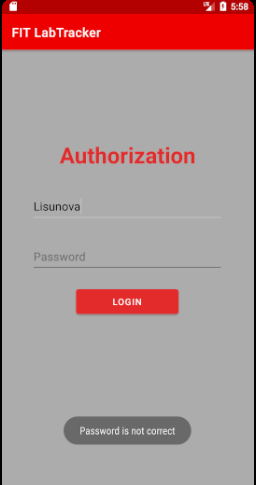


Рисунок 5.1.3 – Сообщение пользователю об ошибке

Также можно проверить случай, когда преподаватель попытается ввести количество сданных лабораторных бльше общего количество или меньше 0, тогда, появятся соответствующие уведомления

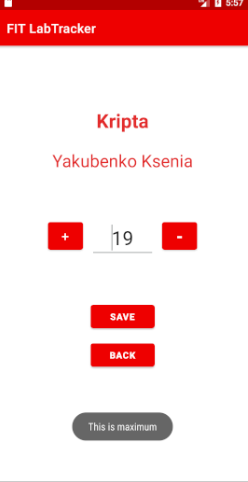
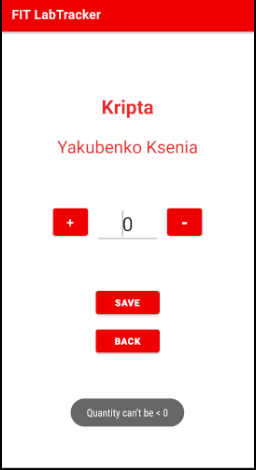


Рисунок 5.1.4– Сообщение пользователю об ошибке

Заключение

При работе над курсовым проектом было проведено изучение предметной области, выявление в ней проблем и составление функционала программного средства, с их учётом, после чего последовало проектирование и разработка. Также в ходе работы проанализировано и проработано множество сторон и нюансов проектирования и разработки мобильного приложения, сервера, а также веб-приложения, например, выбор подходящих технологий, составление и тестирование алгоритмов, структурирование проекта, построение пользовательского интерфейса, синхронизации данных и т.д.

В итоге были реализован весь заявленный функционал и готовое приложение вполне можно назвать подходящим для реального использования. Было получено большое количество опыта по работе со всеми используемыми технологиями и фреймворками.

Список литературы

1. REST-API [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/483202/>. – Дата доступа: 10.04.2021.
2. StudFiles [Электронный ресурс]. – Режим доступа: https://studfiles.net/preview/page:2/. – Дата доступа: 12.04.2021.
3. Трехуровневая архитектура — Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Трехуровневая\_архитектура. – Дата доступа: 19.03.2021.
4. Документация по Android [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://developers.google.com/android/> – Дата доступа: 06.03.2021;
5. Мартин, Р. Чистая архитектура и искусство разработки программного обеспечения / Р. Мартин. – Санкт-Петербург: Питер, 2019. – 352 с
6. Metanit Java [Электронный ресурс]. – Режим доступа: <https://metanit.com/java/android/>. – Дата доступа: 16.05.2021
7. Магазин приложений Apple Store [Электронный ресурс]. – Режим доступа: https://apps.apple.com/us/app/my-life-my-voice-mood-journal/id626899759. – Дата доступа: 13.03.2021
8. Магазин приложений Google Play [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=pl.tlasica.moodtracker&hl=ru&gl=US. – Дата доступа: 12.03.2021
9. Веб-сервис для хостинга [Электронный ресурс]. – Режим доступа: https://github.com/blackfizz/EazeGraph. – Дата доступа: 15.02.2021
10. Веб-сервис для хостинга [Электронный ресурс]. – Режим доступа: https://github.com/FasterXML/jackson. – Дата доступа: 15.02.2021

Приложение A

**Logic Plan Pass add**int all\_quantity\_od\_days\_to\_pass = date\_passes.size() - 1;  
  
if (all\_quantity\_od\_days\_to\_pass / all\_quantity\_of\_lab > 0.5 && all\_quantity\_od\_days\_to\_pass / all\_quantity\_of\_lab <= 1) {  
  
 double quantity = all\_quantity\_of\_lab;  
 for (Date date1 : date\_passes) {  
  
 int Quantity\_to\_pass = 1;  
 SimpleDateFormat format1 = new SimpleDateFormat("dd.MM.yyyy");  
 String Date\_pass = format1.format(date1);  
 int IDLab\_pass = ID\_Lab\_for\_pass;  
  
 db.execSQL("insert into PlansPass( IDLab, Date, Quantity) VALUES("  
 + IDLab\_pass + ",'" + Date\_pass  
 + "'," + Quantity\_to\_pass + ")"  
 );  
  
 g = 1;  
 quantity--;  
 }  
 while (quantity > 0) {  
 all\_quantity\_od\_days\_to\_pass = date\_passes.size() - 1;  
 List<Date> date\_change\_1 = date\_passes;  
  
 if (all\_quantity\_od\_days\_to\_pass / quantity > 1) {  
 int day\_plus = (int) Math.round(all\_quantity\_od\_days\_to\_pass / quantity);  
 if (day\_plus > 0) {  
 day\_plus--;  
 }  
 int previous\_3 = 0;  
 for (int k = 0; k < date\_change\_1.size(); k++) {  
 if (quantity > 0) {  
 previous\_3 = previous\_3 + (day\_plus);  
 if (previous\_3 < date\_change\_1.size()) {  
 SimpleDateFormat format1 = new SimpleDateFormat("dd.MM.yyyy");  
 String Date\_pass = format1.format(date\_change\_1.get(previous\_3));  
  
 int IDLab\_pass = ID\_Lab\_for\_pass;  
  
 Cursor getQuantityByDate = db.rawQuery("Select Quantity from PlansPass where IDLab=" + ID\_Lab\_for\_pass + " AND Date ='" + Date\_pass + "' LIMIT 1", null);  
 int i = 0;  
  
 if (getQuantityByDate.moveToFirst()) {  
 plan = new PlansPass(getQuantityByDate.getInt(0));  
 }  
 int Quantity\_to\_pass = plan.Quantity;  
  
 Quantity\_to\_pass = Quantity\_to\_pass + 1;  
  
 db.execSQL("update PlansPass set Quantity=" + Quantity\_to\_pass  
 + " WHERE Date='" + Date\_pass  
 + "' AND IDLab=" + IDLab\_pass  
 );  
 g=1;  
 quantity--;  
 date\_passes.remove(previous\_3);  
 }  
 }  
 }  
  
 } else if (all\_quantity\_od\_days\_to\_pass / quantity > 0.5 && all\_quantity\_od\_days\_to\_pass / quantity <= 1) {  
 for (Date date1 : date\_passes) {  
 find\_plan\_pass\_for\_date = db.rawQuery("Select Quantity from PlansPass WHERE Date='" + date1.toString() + "' and IDLab=" + ID\_Lab\_for\_pass, null);  
 if (find\_plan\_pass\_for\_date.moveToFirst()) {  
 while (!find\_plan\_pass\_for\_date.isClosed()) {  
 plan\_pass\_for\_one\_of\_date = new PlansPass(find\_plan\_pass\_for\_date.getInt(0));  
 if (!find\_plan\_pass\_for\_date.isLast()) {  
 find\_plan\_pass\_for\_date.moveToNext();  
 } else {  
 find\_plan\_pass\_for\_date.close();  
 }  
 }  
 }  
 int Quantity\_to\_pass = plan\_pass\_for\_one\_of\_date.Quantity + 1;  
  
 SimpleDateFormat format1 = new SimpleDateFormat("dd.MM.yyyy");  
 String Date\_pass = format1.format(date1);  
  
 int IDLab\_pass = ID\_Lab\_for\_pass;  
  
 db.execSQL("update PlansPass set Quantity =" + Quantity\_to\_pass  
 + " WHERE Date='" + Date\_pass + "' and IDLab=" + IDLab\_pass  
 );  
  
 g=1;  
 quantity--;  
 }  
  
 }  
 }  
} else if (all\_quantity\_od\_days\_to\_pass / all\_quantity\_of\_lab <= 0.5) {  
 double quantity = all\_quantity\_of\_lab;  
 for (Date date1 : date\_passes) {  
 if (quantity > 1) {  
 int Quantity\_to\_pass = 2;  
  
 SimpleDateFormat format1 = new SimpleDateFormat("dd.MM.yyyy");  
 String Date\_pass = format1.format(date1);  
  
 int IDLab\_pass = ID\_Lab\_for\_pass;  
  
 db.execSQL("insert into PlansPass( IDLab, Date, Quantity) VALUES("  
 + IDLab\_pass + ",'" + Date\_pass  
 + "'," + Quantity\_to\_pass + ")"  
 );  
  
  
 g = 1;  
 quantity = quantity - 2;  
 }  
 }  
 while (quantity > 0) {  
 all\_quantity\_od\_days\_to\_pass = date\_passes.size() - 1;  
 List<Date> date\_change\_1 = date\_passes;  
  
 if (all\_quantity\_od\_days\_to\_pass / quantity > 1) {  
 int day\_plus = (int) Math.round(all\_quantity\_od\_days\_to\_pass / quantity);  
 if (day\_plus > 0) {  
 day\_plus--;  
 }  
 int previous\_3 = 0;  
 for (int k = 0; k < date\_change\_1.size(); k++) {  
 if (quantity > 0) {  
  
 previous\_3 = previous\_3 + (day\_plus);  
 if (previous\_3 < date\_change\_1.size()) {  
  
 SimpleDateFormat format1 = new SimpleDateFormat("dd.MM.yyyy");  
 String Date\_pass = format1.format(date\_change\_1.get(previous\_3));  
  
  
 int IDLab\_pass = ID\_Lab\_for\_pass;  
  
 Cursor getQuantityByDate = db.rawQuery("Select Quantity from PlansPass where IDLab=" + ID\_Lab\_for\_pass + " AND Date ='" + Date\_pass + "' LIMIT 1", null);  
 int i = 0;  
  
 if (getQuantityByDate.moveToFirst()) {  
 plan = new PlansPass(getQuantityByDate.getInt(0));  
 }  
 int Quantity\_to\_pass = plan.Quantity;  
  
 Quantity\_to\_pass = Quantity\_to\_pass + 1;  
  
 db.execSQL("update PlansPass set Quantity=" + Quantity\_to\_pass  
 + " WHERE Date='" + Date\_pass  
 + "' AND IDLab=" + IDLab\_pass  
 );  
  
 g=1;  
  
 quantity--;  
  
 date\_passes.remove(previous\_3);  
  
 }  
 }  
  
 }  
 }  
 }  
} else if (all\_quantity\_od\_days\_to\_pass / all\_quantity\_of\_lab > 1) {  
 double quantity = all\_quantity\_of\_lab;  
 int day\_plus = (int) Math.round(all\_quantity\_od\_days\_to\_pass / all\_quantity\_of\_lab);  
 if (day\_plus > 0) {  
 day\_plus--;  
 }  
 List<Date> date\_change = date\_passes;  
 int previous\_1 = 0;  
 for (int i = 0; i < date\_change.size(); i++) {  
 if (quantity > 0) {  
 previous\_1 = previous\_1 + (day\_plus);  
 if (previous\_1 < date\_change.size()) {  
 int Quantity\_to\_pass = 1;  
  
 SimpleDateFormat format1 = new SimpleDateFormat("dd.MM.yyyy");  
 String Date\_pass = format1.format(date\_change.get(previous\_1));  
  
 int IDLab\_pass = ID\_Lab\_for\_pass;  
  
 db.execSQL("insert into PlansPass( IDLab, Date, Quantity) VALUES("  
 + IDLab\_pass + ",'" + Date\_pass  
 + "'," + Quantity\_to\_pass + ")"  
 );  
 g = 1;  
  
 quantity--;  
 date\_passes.remove(previous\_1);  
  
 }  
 }  
 }  
 while (quantity > 0) {  
 all\_quantity\_od\_days\_to\_pass = date\_passes.size() - 1;  
 List<Date> date\_change\_1 = date\_passes;  
  
 if (all\_quantity\_od\_days\_to\_pass / quantity > 1) {  
 day\_plus = (int) Math.round(all\_quantity\_od\_days\_to\_pass / quantity);  
 if (day\_plus > 0) {  
 day\_plus--;  
 }  
  
 int previous\_3 = 0;  
 for (int k = 0; k < date\_change\_1.size(); k++) {  
 if (quantity > 0) {  
  
 previous\_3 = previous\_3 + (day\_plus);  
 if (previous\_3 < date\_change\_1.size()) {  
  
 SimpleDateFormat format1 = new SimpleDateFormat("dd.MM.yyyy");  
 String Date\_pass = format1.format(date\_change\_1.get(previous\_3));  
  
  
 int IDLab\_pass = ID\_Lab\_for\_pass;  
  
 Cursor getQuantityByDate = db.rawQuery("Select Quantity from PlansPass where IDLab=" + ID\_Lab\_for\_pass + " AND Date ='" + Date\_pass + "' LIMIT 1", null);  
 int i = 0;  
  
 if (getQuantityByDate.moveToFirst()) {  
 plan = new PlansPass(getQuantityByDate.getInt(0));  
 }  
 int Quantity\_to\_pass = plan.Quantity;  
  
 Quantity\_to\_pass = Quantity\_to\_pass + 1;  
  
 db.execSQL("update PlansPass set Quantity=" + Quantity\_to\_pass  
 + " WHERE Date='" + Date\_pass  
 + "' AND IDLab=" + IDLab\_pass  
 );  
  
 g=1;  
  
 quantity--;  
  
 date\_passes.remove(previous\_3);  
  
  
 }  
  
 }  
  
  
 }  
  
 }  
  
 }  
  
}

Приложение Б

public interface JSONPlaceHolderApi {  
 @GET("Students/{login}")  
 public Call<LoginStudents> getStudentByLogin(@Path("login") String Login);  
 @GET("Teachers/{login}")  
 public Call<LoginTeachers> getTeacherByLogin(@Path("login") String Login);  
 @GET("ListStudents/{id}")  
 public Call<Students> getStudent(@Path("id") int IDStudent);  
 @GET("ListStudents")  
 public Call<List<Students>> getAllStudents();  
 @GET("ListTeachers/{id}")  
 public Call<Teachers> getTeacher(@Path("id") int IDTeacher);  
 @GET("Students")  
 public Call<List<LoginStudents>> getStudentByID();  
 @GET("Teachers")  
 public Call<List<LoginTeachers>> getTeacherByID();  
 @POST("Students")  
 public Call<LoginStudents> addStudent(@Body LoginStudents Student);  
 @POST("Teachers")  
 public Call<LoginTeachers> addTeacher(@Body LoginTeachers Teacher);  
 @GET("Groups/{id}")  
 public Call<Group> getGroup(@Path("id") int IDGroup);  
 @GET("Cources/{id}")  
 public Call<Course> getCourse(@Path("id") int IDCource);  
 @PUT("Students/{login}")  
 public Call<LoginStudents> updateStudent(@Body LoginStudents Student, @Path("login") String Login);  
 @PUT("Teachers/{login}")  
 public Call<LoginTeachers> updateTeacher(@Body LoginTeachers Teacher, @Path("login") String Login);  
 @GET("Semestrs/{id}")  
 public Call<Semesters> getSemesterByID(@Path("id") int IDSem);  
 @GET("ListLabs")  
 public Call<List<ListLabs>> getListLab();  
 @GET("ListLabs/{id}")  
 public Call<ListLabs> getListLabByID(@Path("id") int IDLab);  
 @GET("ListLabTeachers")  
 public Call<List<ListLabTeachers>> getListLabTeachers();  
 @GET("Semestrs")  
 public Call<List<Semesters>> getSemesterAll();  
 @GET("StudentPasses")  
 public Call<List<StudentPassess>> getStudentPasses();  
 @POST("StudentPasses")  
 public Call<StudentPassess> addStudentPasses(@Body StudentPassess Pass);  
 @POST("StudentPasses/{id}")  
 public Call<StudentPassess> updateStudentPasses( @Path("id") int IDStudentPass, @Body StudentPassess Pass);  
  
  
  
  
}

Приложение В

public class ListStudentsController : ApiController

{

private LabTrackerDB db = new LabTrackerDB();

// GET: api/ListStudents

public IQueryable<ListStudent> GetListStudents()

{

db.Configuration.ProxyCreationEnabled = false;

return db.ListStudents;

}

// GET: api/ListStudents/5

[ResponseType(typeof(ListStudent))]

public IHttpActionResult GetListStudent(int id)

{

db.Configuration.ProxyCreationEnabled = false;

ListStudent listStudent = db.ListStudents.Find(id);

if (listStudent == null)

{

return NotFound();

}

return Ok(listStudent);

}

// PUT: api/ListStudents/5

[ResponseType(typeof(void))]

public IHttpActionResult PutListStudent(int id, ListStudent listStudent)

{

db.Configuration.ProxyCreationEnabled = false;

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

if (id != listStudent.IDStudent)

{

return BadRequest();

}

db.Entry(listStudent).State = EntityState.Modified;

try

{

db.SaveChanges();

}

catch (DbUpdateConcurrencyException)

{

if (!ListStudentExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return StatusCode(HttpStatusCode.NoContent);

}

// POST: api/ListStudents

[Authorize]

[ResponseType(typeof(ListStudent))]

public IHttpActionResult PostListStudent(ListStudent listStudent)

{

db.Configuration.ProxyCreationEnabled = false;

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

db.ListStudents.Add(listStudent);

try

{

db.SaveChanges();

}

catch (DbUpdateException)

{

if (ListStudentExists(listStudent.IDStudent))

{

return Conflict();

}

else

{

throw;

}

}

return CreatedAtRoute("DefaultApi", new { id = listStudent.IDStudent }, listStudent);

}

// DELETE: api/ListStudents/5

[Authorize]

[ResponseType(typeof(ListStudent))]

public IHttpActionResult DeleteListStudent(int id)

{

db.Configuration.ProxyCreationEnabled = false;

ListStudent listStudent = db.ListStudents.Find(id);

if (listStudent == null)

{

return NotFound();

}

db.ListStudents.Remove(listStudent);

db.SaveChanges();

return Ok(listStudent);

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

private bool ListStudentExists(int id)

{

return db.ListStudents.Count(e => e.IDStudent == id) > 0;

}

}

ПРИЛОЖЕНИЕ Г

@section scripts

{

<script type="text/javascript">

$(document).ready(function () {

$("#addTeacher").click(function (event) {

AddTeacher();

});

});

// Добавление новой книги

function AddTeacher() {

// получаем значения для добавляемой книги

var teacher = {

IDTeacher: $('#addID').val(),

Name: $('#addName').val(),

Surname: $('#addSurname').val()

};

$.ajax({

url: '/api/ListTeachers/',

type: 'POST',

data: JSON.stringify(teacher),

contentType: "application/json;charset=utf-8",

success: function (data) {

window.location = 'https://localhost:44350/Home/ListOfTeachers';

},

error: function (x, y, z) {

alert('Error' + '\n' +'Pleace checked the fields '+ '\n' + 'End try again');

}

});

}

</script>

}