МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ижевский государственный технический университет имени М. Т. Калашникова»

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение»

Работа защищена с оценкой

«\_\_\_\_»

Дата \_\_\_\_\_\_\_\_\_\_

Подпись \_\_\_\_\_\_\_/\_\_\_\_\_\_\_

ОТЧЕТ ПО ПРЕДДИПЛОМНОЙ ПРАКТИКЕ

«Система управления учебным процессом. Мобильный клиент»

Выполнил:

Студент группы Б08-191-2 И.О. Зырянов

Руководитель:

к.т.н., доцент кафедры ПО И.О. Архипов

Рецензия:

степень достижения поставленной цели работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

полнота разработки темы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

уровень самостоятельности работы обучающегося\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

недостатки работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

СОДЕРЖАНИЕ

[2. РАЗРАБОТКА ЗАДАЧ ДЛЯ МОБИЛЬНОГО КЛИЕНТА 3](#_Toc41012537)

[2.1. Разработка модуля авторизации 3](#_Toc41012538)

[2.1.1. Описание постановки задачи 3](#_Toc41012539)

[2.1.2. Описание алгоритма регистрации пользователей 4](#_Toc41012540)

[2.1.3 Описание алгоритма авторизации 6](#_Toc41012541)

2. РАЗРАБОТКА ЗАДАЧ ДЛЯ МОБИЛЬНОГО КЛИЕНТА

2.1. Разработка модуля авторизации

2.1.1. Описание постановки задачи

2.1.1.1. Характеристика задачи

Задача – разработать модуль авторизации в мобильном клиенте.

Модуль авторизации необходим для выдачи доступа пользователям, которые будут использовать данные приложения.

Разработка модуля авторизации связана с разработкой компонента навигации приложения и реализацией взаимодействия приложения с сервисом API.

Разработка задачи будет завершена после успешной реализации двух компонентов модуля: регистрации и аутентификации.

2.1.1.2. Входная информация

Перечень входных данных при регистрации пользователей:

1. номер зачетной книжки (в случае, если пользователь – студент);
2. идентификатор преподавателя в системе (в случае, если пользователь – преподаватель);
3. уникальный логин пользователя;
4. пароль.

Перечень входных данных при аутентификации и авторизации пользователей:

1. логин пользователя, созданный при регистрации;
2. пароль пользователя, созданный при регистрации.

Вся вышеперечисленная информация получается непосредственно через ручной ввод данных пользователем, а также посредством получения данных от сервера с помощью http запросов.

2.1.1.3. Выходная информация

Перечень выходных данных при регистрации пользователей:

1. сообщение об успешной или о неудачной регистрации;
2. интент для перехода на следующий экран приложения.

Перечень выходных данных при аутентификации и авторизации пользователей:

1. тип доступа;
2. интент для перехода на следующий экран приложения;
3. Сообщение об ошибке при некорректных данных/ошибке сервера

Вся вышеперечисленная информация отображается и передается приложением в ответ на http-запросы.

2.1.2. Описание алгоритма регистрации пользователей

2.1.2.1. Назначение и характеристика

Алгоритм регистрации необходим для передачи регистрационных данных пользователя в базу данных и получения ответа со статусом регистрации с последующим переходом к экрану авторизации.

Алгоритм должен отправлять регистрационные данные пользователя на сервер и, при получения данных с сервера, обрабатывать их с последующим переходом к другим модулям приложения, в зависимости от обработанных данных.

2.1.2.2. Используемая информация

Для алгоритма регистрации используются:

1. номер зачетной книжки студента или идентификатор преподавателя в системе;
2. уникальный логин пользователя;
3. пароль пользователя.

2.1.2.3. Результаты решения

В результате выполнения алгоритма формируется следующая информация, в зависимости от входной информации:

1. интент для перехода на следующий экран приложения – при успешном ответе от сервера;
2. сообщение об ошибке при регистрации/аутентификации – при ошибке.

2.1.2.4. Алгоритм решения

Схема алгоритма представлена на изображении ниже (рис. 1)



Рис. 1. Схема работы компонента регистрации клиента

Описание алгоритма:

1. Пользователь вводит регистрационные данные в поля ввода на экране приложения
2. Эти данные – уникальный логин, пароль и идентификатор пользователя (номер зачетки для студента) – сохраняются в локальном хранилище с помощью использования класса регистрации пользователя UserRegistration().
3. Формируется POST запрос с использованием объектов класса (для передачи регистрационных данных в body запроса)
4. В ответ на запрос сервер присылает код ответа. Этот ответ преобразуется из формата Json в объект класса и анализируется.
5. Если код – ошибка, то она отображается на экране. После этого поля ввода данных на экране очищаются
6. Если код сигнализирует об успешной регистрации – формируется интент для перехода на экран авторизации.

2.1.3 Описание алгоритма авторизации

2.1.3.1. Назначение и характеристика

Алгоритм авторизации необходим для передачи на сервер авторизационных данных и получения токенов доступа, по которым определяется доступ пользователя к ресурсам и функциям клиентского приложения.

В случае первой авторизации – необходима авторизация по логину и паролю. При повторном логине используется токен авторизации, а сам логин происходит с помощью ввода ПИН-кода.

2.1.3.2. Используемая информация

Для алгоритма авторизации используются:

1. логин пользователя;
2. пароль пользователя;
3. токен авторизации пользователя.

2.1.3.3. Результаты решения

В результате выполнения алгоритма формируется следующая информация, в зависимости от приходящей от сервера информации:

1. интент для перехода на следующий экран приложения – при успешном ответе от сервера;
2. сообщение об ошибке при авторизации – при ошибке.

2.1.2.4. Алгоритм решения

Схемы алгоритма представлены на изображении ниже (рис. 2,3)

Рис. 1. Схема авторизации по ПИН-коду Рис. 2. Схема авторизации по логину и паролю

* 1. Разработка навигационного модуля приложения
     1. Описание постановки задачи
        1. Характеристика задачи

Задача – разработать модуль навигации в приложении.

Модуль навигации необходим для определения переходов и связей между различными компонентами и экранами (фрагментами) приложения.

Разработка архитектуры навигации связана с разработкой отдельных экранов приложения и реализацией отображения в них данных, получаемых с сервера.

Разработка задачи будет завершена после успешной реализации переходов между основными экранами приложения.

* + - 1. Входная информация

1. Ресурсный идентификатор текущего экрана
2. Интент перехода
3. Тип доступа клиента

Информация о текущем экране получается при запуске приложения и при переходе по навигационному графу приложения.

* + - 1. Выходная информация

1. Ресурсный идентификатор нужного экрана

Данная информация используется для отображения данных, получаемых с сервера, на конкретном экране.

* + 1. Описание алгоритма навигации
       1. Назначение и характеристика

Алгоритм навигации между фрагментами необходим для реализации переходов между ними с использованием входных данных.

Алгоритм должен производить переход между фрагментами заранее созданной активности с использованием контейнера NavHostFragment, управляющим перемещением в destinations – точки назначения.

Активити используется как основная точка входа в приложение, в то время как за переходы между отдельными фрагментами отвечает NavHostFragment.

Для каждого фрагмента в отдельности реализован kotlin-класс фрагмента – для управления данными внутри него. Как только модуль навигации меняет текущий фрагмент, данные автоматически начинают обрабатываться в этом классе.

Так же необходимо реализовать NavController – метод, ассоциируемый с NavHostFragment, с помощью которого реализуются сами переходы.

* + 1. Алгоритм решения

Схема алгоритма:

**Здесь будет алгоритм связи между активити, NavHostFragment, NavController и самим фрагментом**

Описание алгоритма:

**Здесь будет описание алгоритма связи между активити, NavHostFragment, NavController и самим фрагментом**

**Здесь будет Схема навигационного графа всего приложения?**

* 1. Разработка модуля отображения и кэширования данных в отдельных фрагментах.
     1. Описание постановки задачи
        1. Характеристика задачи

Задача – разработать систему получения и отображения данных в приложении посредством использования API-интерфейса, передающего эти данные клиенту из БД с помощью http-запросов. Так же необходимо реализовать методы кэширования этих данных. Разработка методов кеширования необходима для увеличения эффективности работы приложения и улучшения пользовательского опыта.

Разработка модуля связана в первую очередь с отображением и переходом на отдельные фрагменты, то есть непосредственно с архитектурой навигации приложения.

Разработка задачи будет завершена после успешной реализации отображения и кэширования данных на экране приложения, полученных от сервера.

* + - 1. Входная информация

Входная информация передается в виде ответа вида json на http-запросы, отправляемые клиентом. Данные, отображаемые в приложении зависят как от самого запроса, так и от возвращаемых сервером данных. Пример входной информации представлен на изображении ниже (рис. Х)

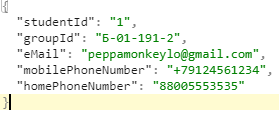


Рис. Х. Ответ на запрос api/university/user-student-requisites

* + - 1. Выходная информация

Выходной информацией будут данные, полученные из запроса и сохраненные в виде класса данных клиента, который работает в виде хранилища данных.

* + 1. Описание алгоритма обработки и отображения данных.
       1. Назначение и характеристика

Алгоритм обработки и отображения данных необходим для корректного формирования конкретного экрана пользовательского приложения. В зависимости от приходящей в запросе информации, данные должны динамически отображаться в выбранном фрагменте клиента.

Алгоритм должен формировать запрос к API с помощью библиотеки Retrofit. При этом получаемая информация должна быть сериализована из json-формата и передана на экран.

* + - 1. Используемая информация

Для формирования запроса необходим идентификатор пользователя.

Идентификатор формируется на этапе авторизации пользователя и хранится во внутреннем хранилище клиента. В зависимости от типа клиента – информация может запрашиваться как для студента, так и для преподавателя.

Далее - используемая информация (данные, получаемые от сервера) полностью зависит от типа запроса.

* + - 1. Результаты решения

В результате выполнения алгоритма данные, сохраненные в классе данных клиента, однозначно определяются к элементам выбранного экрана. Далее происходит формирование и отображение полноценного экрана из отдельных элементов с данными внутри.

* + - 1. Алгоритм решения

1. Формируется и выполняется GET запрос с передачей идентификатора пользователя. Запрос формируется с помощью встроенных методов библиотеки Retrofit
2. В ответ на запрос приходят данные в формате Json. В случае ошибки при получении данных, приходит код ошибки с текстом. Ошибка может возникнуть в случае передачи некорректного параметра, либо при ошибках в работе сервера
3. Полученный Json десериализуем в объекты
4. Если во время запроса была получена ошибка, отображаем текст ошибки на экране и заново формируем запрос
5. Если ошибки не было, сохраняем полученные объекты с помощью Data class’a – специального класс, в котором хранятся данные пользователя
6. Отображаем данные из Data class’a в выбранном фрагменте приложения.

Схематичная работа алгоритма представлена на изображении ниже (рис. Х)



Рис. X. Схема алгоритма загрузки и отображения данных