Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра «Информатика»

УТВЕРЖДАЮ
Заведующий кафедрой

_____ А. С. Кузнецов
подпись инициалы, фамилия

« ___ » _____ 2021 г

БАКАЛАВРСКАЯ РАБОТА

09.03.04 Программная инженерия

Разработка back-end части модуля анализа успеваемости студентов на основе СЭО СФУ

Руководитель ВКР _____ доцент, канд. техн. наук А. В. Хныкин
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ Д. О. Шкуринский
подпись, дата инициалы, фамилия

Красноярск 2021

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка back-end части модуля анализа успеваемости студентов на основе СЭО СФУ» содержит 40 страниц текстового документа, 21 использованный источник, 16 иллюстраций.

ВЕБ-ПРИЛОЖЕНИЕ, BACK-END, ВЕБ-ПЛАТФОРМА, АНАЛИЗ УСПЕВАЕМОСТИ.

Целью данной выпускной квалификационной работы является разработка back-end части модуля анализа успеваемости студентов на основе системы электронного образования СФУ. Разработанное веб-приложение позволяет родителям просматривать учебную информацию в удобном и лаконичном виде, а преподавателям просматривать и анализировать статистику по своим предметам, что должно помочь с определением слабых мест в образовательном процессе.

Для достижения цели были решены следующие задачи:

– проанализированы существующие решения;

– рассмотрены технические средства разработки и определён технологический стек;

– спроектированы архитектура системы и архитектура базы данных.

В результате исследования предметной области были проанализированы существующие решения и был сделан вывод, что в данный момент информация показывается в виде таблиц со множественным уровнем вложенности. Проанализировать аналоги не было возможности, так техническая информация о них отсутствует в свободном доступе, и для работы с ними необходимо являться работником, либо студентом конкретного учебного заведения. Были составлены функциональные требования к приложению.

В итоге была разработана back-end часть приложения со следующими возможностями:

– авторизация в системе;

– просмотр родителями учебной информации своего ребёнка;

– просмотр преподавателями агрегированного отчёта по выбранным группам;

– просмотр преподавателями отчёта по каждой выбранной группе отдельно.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Длящаяся по сей день пандемия COVID-19 и последующий массовый переход на дистанционный режим обучения поставили перед учебными заведениями ряд трудностей, в частности усложнился процесс контроля над успеваемостью обучающихся. Также новый режим обучения породил необходимость оценки его эффективности. Данные проблемы могут быть решены с помощью системы анализа успеваемости учащихся.

Цель данной выпускной квалификационной работы – разработка back-end части веб-приложения, предоставляющего подробную статистику по успеваемости студентов ИКИТ.

Функционал веб-приложения предназначен для родителей студентов и преподавателей: родители смогут просматривать успеваемость и посещаемость занятий своего ребёнка, преподаватели получат инструменты для сравнения успеваемости студентов по преподаваемым дисциплинам в учебных группах по отдельности и вместе.

Для достижения цели были реализованы следующие задачи:

– проведён анализ существующих систем анализа успеваемости студентов;

– исследована предметная область;

– создана модель предметной области;

– спроектирована архитектура back-end части приложения;

– спроектирована база данных;

– реализована back-end часть приложения;

– проведено тестирование.

1 Анализ предметной области

Большинство существующих систем анализа успеваемости, предоставляемых преподавателям и студентам в учебных заведениях, являются не более чем набором таблиц. Высокий уровень вложенности и однообразие затрудняют пользование подобными системами.

Предлагаемое веб-приложение даст возможность родителям просматривать информацию об успеваемости их ребёнка, а преподавателям – просматривать информацию об успеваемости учебных групп по их предметам в упорядоченном и лаконичном виде.

Помимо анализа предметной области, также необходимо выбрать стек технологий, который будет использоваться в разработке.

1.1 Анализ существующих решений

Системы отслеживания успеваемости можно условно разделить на две категории: комплексные готовые решения, которые предлагают также отслеживание оплаты за обучение, например, продукты Creatrix Campus [14], Engage [20] и eduCloud [19], и (предположительно) самодельные системы, которыми пользуются, к примеру, Томский государственный университет систем управления и радиоэлектроники (ТУСУР), Казанский федеральный университет и Южный институт менеджмента (ЮИМ). К сожалению, чтобы получить демонстрационную версию готового решения, требуется быть сотрудником образовательного учреждения. Доступ к системам указанных выше вузов также затруднён, так как даётся лишь их студентам. Данные обстоятельства и отсутствие технической информации об этих системах в открытом доступе делают анализ невозможным.

1.2 Определение требований к системе

Ввиду невозможности анализа аналогов, источниками требований стали пожелания родителей и также ограничения АСУ ИКИТ и СЭО СФУ, а также пожелания преподавателей, появившиеся в процессе разработки:

– авторизация в системе должна осуществляться посредством логина и пароля от аккаунта СФУ;

– авторизация должна осуществляться через АСУ ИКИТ;

– данные для родителей должны содержать сведения о посещаемости и успеваемости учащегося;

– данные для преподавателей должны включать сведения о посещаемости и успеваемости учебных групп;

– для преподавателей должна быть предусмотрена фильтрация по учебным группам, дисциплинам и годам;

– модуль должен представлять собой самостоятельное приложение: интеграция с АСУ ИКИТ будет проводиться посредством перенаправления с основного сайта.

1.3 Выбор средств разработки

При определении стека технологий встал выбор между двумя парами, с которыми имеется наибольший опыт работы: СУБД InterSystems Cache и язык программирования ObjectScript или СУБД PostgreSQL и язык программирования Java.

InterSystems Cache [7] – объектно-ориентированная СУБД, позиционируемая производителем как «постреляционная» и мультимодельная, разработана в 1997 году компанией InterSystems. Её неотъемлемой частью является язык ObjectScript: надстройка над языком M стандарта ISO 11756-1999 с поддержкой объектно-ориентированного программирования и встроенного SQL [12]. Важным преимуществом ObjectScript над другими языками является то, что так называемые персистентные (постоянные) классы описывают структуру хранимых данных и позволяют значительно упростить работу с ними, устраняя необходи-

мость в создании транспортных объектов и логики для промежуточной обработки данных

Также вместе с Cache поставляется среда разработки Studio.

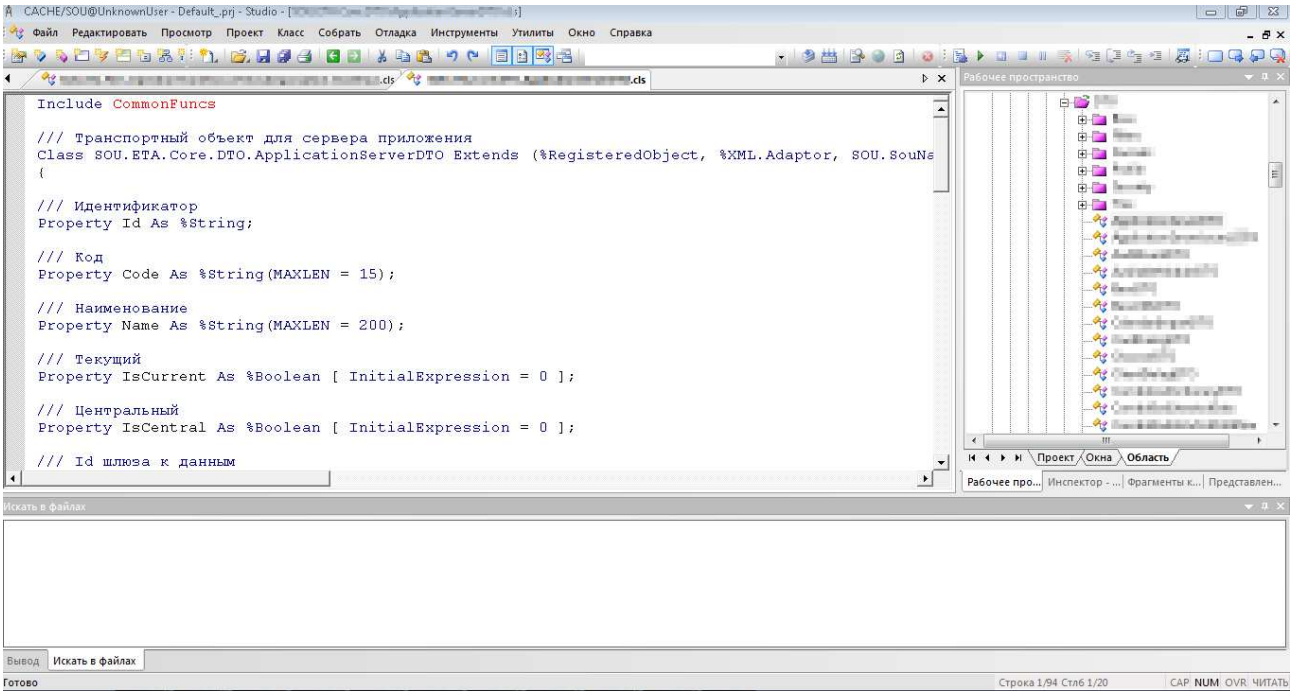


Рисунок 1 – рабочая область Studio

Несмотря на все достоинства Cache, возможные затруднения в последующем сопровождении другим разработчиком из-за относительной малоизвестности Cache и проприетарная лицензия заставили отказаться от этого варианта. Таким образом, выбор пал на Java и PostgreSQL.

Java – строго типизированный, объектно-ориентированный язык программирования, разработанный в 1995 году компанией Sun Microsystems. Его преимущества:

– использование виртуальной Java-машины (JVM) позволяет легко переносить приложение с одной платформы на другую;

– подробная документация;

– большое количество библиотек и фреймворков, в том числе для веб-разработки.

К недостаткам следует отнести то, что в сравнении с приложениями, реализованными на других языках, Java-приложение может показывать меньшую производительность, что обусловлено использованием виртуальной машины.

При разработке использовалась версия Java 11. Выбор обусловлен тем, что на момент разработки это новейшая LTS-библиотека для Java (от англ. long-term support, долговременная поддержка).

В качестве основного фреймворка был выбран Spring Boot Starter Web. Данный фреймворк позволяет значительно упростить back-end разработку REST-приложения благодаря набору готовых компонентов, встроенному серверу и простому механизму конфигурирования. Большим преимуществом является его широкое распространение, что подразумевает наличие подробной документации и множества обучающих курсов. К недостаткам можно отнести большое количество входящих в его состав лишних зависимостей, которые не будут использоваться приложением, что увеличит его размер.

Дополнительные библиотеки, использованные при разработке:

– Jackson [10] для упрощения работы с форматом данных JSON;

– Lombok [18]. Данная библиотека позволяет избежать написания повторяющегося (boilerplate) кода, например, конструкторов, с помощью использования аннотаций.

В качестве редактора кода была выбрана среда разработки IntelliJ IDEA от компании JetBrains [11]. Включает в себя отладчик, инструменты для работы с Git, поддержку плагинов, подсветку синтаксиса и средства для рефакторинга. Для работы была выбрана бесплатная версия Community.

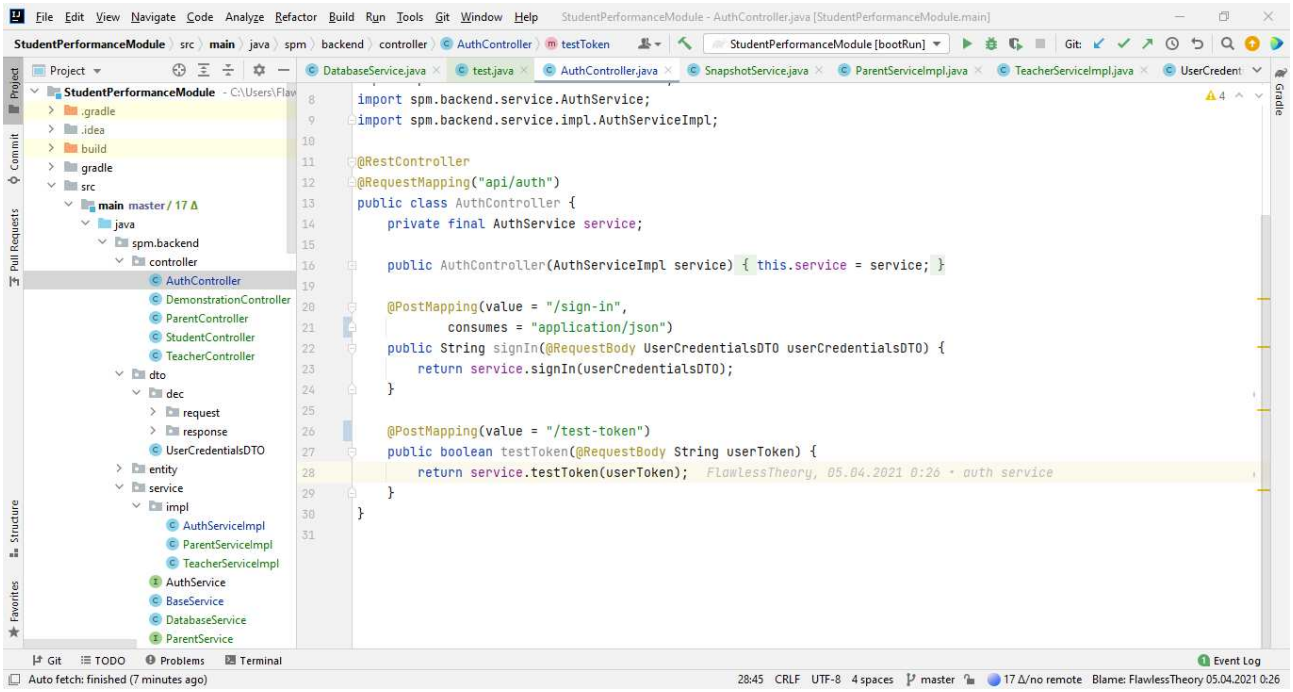


Рисунок 2 – интерфейс IntelliJ IDEA

PostgreSQL [15] – объектно-реляционная СУБД, разработанная в 1996 году на основе некоммерческой СУБД Postgres. Её достоинствами является свободное распространение, высокая производительность и встроенная поддержка слабоструктурированных данных в формате JSON с возможностью их индексации.

При разработке веб-приложения использовалась система контроля версий Git. Репозиторий приложения размещался на хостинге Github. Git позволяет как легко просматривать изменения между версиями, так и быстро перемещаться между ними, совершать откаты, слияния и прочее. В качестве графического клиента для Git был выбран TortoiseGit [21].

Для тестирования API было выбрано ПО Postman [17].

1.4 Методология разработки

Гибкая методология (Agile) разработки является одним из самых популярных подходов к разработке программного обеспечения, так как она весьма эффективна при работе в составе небольшого коллектива. Также она позволяет вносить изменения в систему на любом из этапов разработки. Ввиду того, что в процессе разработки регулярно появлялись новые идеи, именно она была вы-

брана для работы над приложением, точнее, одна из реализаций данного подхода, SCRUM [2]. Основные его принципы:

– работа разбивается на чётко определённые временные промежутки, итерации, в данном случае – по неделям;

– в конце каждой итерации проводится собрание для отслеживания прогресса работ.

Фиксирование и распределение задач проводились в сервисе Trello.

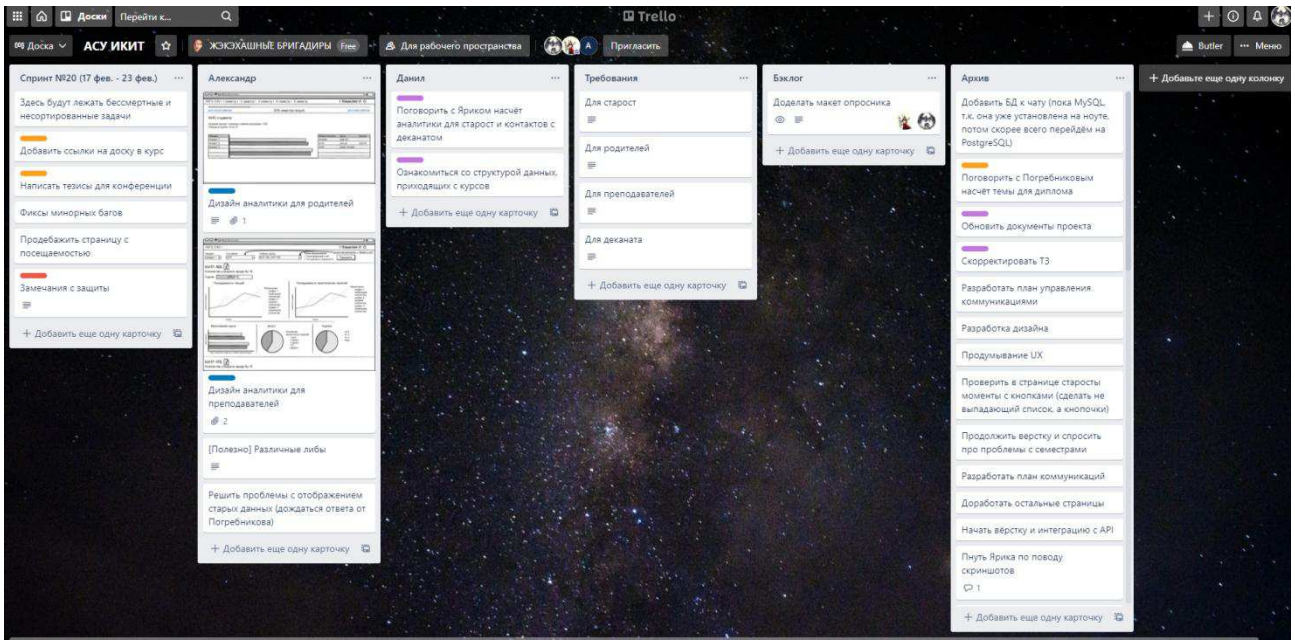


Рисунок 3 – Доска на Trello

1.5 Выводы по разделу

Целью раздела являлось изучение предметной области, в результате которого были собраны основные требования к разрабатываемому веб-приложению. Также был выбран стек технологий для разработки.

2 Проектирование

2.1 Архитектура системы

Архитектура веб-приложения в основном представляет отношения и взаимодействия между такими компонентами, как пользовательские интерфейсы, мониторы обработки транзакций, базы данных и другие. Все веб-

приложения состоят из двух частей – клиентской (front-end) и серверной (back-end), которая также включает в себя базу данных [3, 4]. Клиентская часть отображает пользователю графический интерфейс и реагирует на запросы, серверная часть обрабатывает и отвечает на поступающие запросы.

Данное приложение было построено с использованием паттерна MVC и архитектуры REST.

Паттерн MVC [6] (от англ. model-view-controller, «модель-представление-контроллер») подразумевает разделение кода на три основных слоя:

– модели. Эти компоненты отвечают за представление данных и изменяют своё состояние в соответствии с действиями контроллеров;

– представления. Эти компоненты отвечают за отображение данных пользователю и взаимодействие с ним. Изменяются в соответствии с представляемыми ими моделями;

– контроллеры. Эти компоненты отвечают за управляющую логику приложения, реагируя на действия пользователя и изменяя состояния моделей, связывая таким образом модели и их представления.

Цель подобного разделения – это отделение бизнес-логики приложения от его представления. Таким образом, снижается сцепленность классов и увеличивается процент повторно используемого кода. Стоит отметить, что считается хорошей практикой выносить саму бизнес-логику в сервисы, а внутри контроллеров оставлять вызовы методов соответствующего сервиса.

REST (от англ. Representational State Transfer — «передача состояния представления») представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. Существует пять обязательных к исполнению ограничений по Филдингу, определяющих то, как сервер обрабатывает и отвечает на запросы клиентов. При их соблюдении система приобретает такие положительные свойства как производительность, масштабируемость, простота, способность к изменениям, переносимость, отслеживаемость и надёжность. Список условий:

– приложение придерживается модели «клиент-сервер»;

– отсутствие состояния на стороне сервера: запрос к нему должен быть составлен так, чтобы в нём содержалась вся необходимая для выполнения информация. Состояние сессии при этом сохраняется на стороне клиента;

– кэширование;

– единообразие интерфейса;

– слои. Применение промежуточных серверов способно повысить масштабируемость за счёт балансировки нагрузки и распределённого кэширования.

Теперь рассмотрим непосредственно архитектуру приложения. Фреймворк Spring Web MVC [9], расширением которого является выбранный для разработки Spring Boot Starter Web,

Слой REST-контроллеров представлен тремя компонентами:

– AuthController – ответственен за авторизацию;

– ParentController – обрабатывает запросы, приходящие от родителя;

– TeacherController – обрабатывает запросы от преподавателя.

Для каждого из контроллеров назначен собственный сервис, содержащий необходимую бизнес-логику. DatabaseService – сервис для работы с базой данных. Назначение SnapshotService раскрыто ниже.

Отдельного упоминания требует появившийся во время разработки компонент, который не вписывается в MVC – планировщик обновлений.

При проектировании функционала для преподавателей и доработке функционала для родителей пришлось столкнуться с двумя ограничениями:

– во-первых, хранящихся на стороне АСУ ИКИТ данных недостаточно для того, чтобы удовлетворить выставленным требованиям. Данные об оценках каждого студента в отдельности, необходимые для построения диаграмм успеваемости внутри группы, придётся брать напрямую из СЭО СФУ;

– во-вторых, существует предел количества запросов в единицу времени к API СЭО СФУ, при превышении которого доступ к API для приложения-нарушителя будет временно закрыт. Учитывая то, что все запросы приложения

будут считаться запросами от АСУ ИКИТ, превышение предела может привести к сбоям в её работе, что является абсолютно неприемлемым.

Таким образом, было решено использовать снапшоты (слепки) успеваемости студентов, формируемые по определённому расписанию. Планировщик обновлений – это компонент системы, который запускает процесс формирования снапшотов с использованием SnapshotService, совершающего запросы к СЭО СФУ. После того, как снапшот был сформирован, он записывается в базу данных.

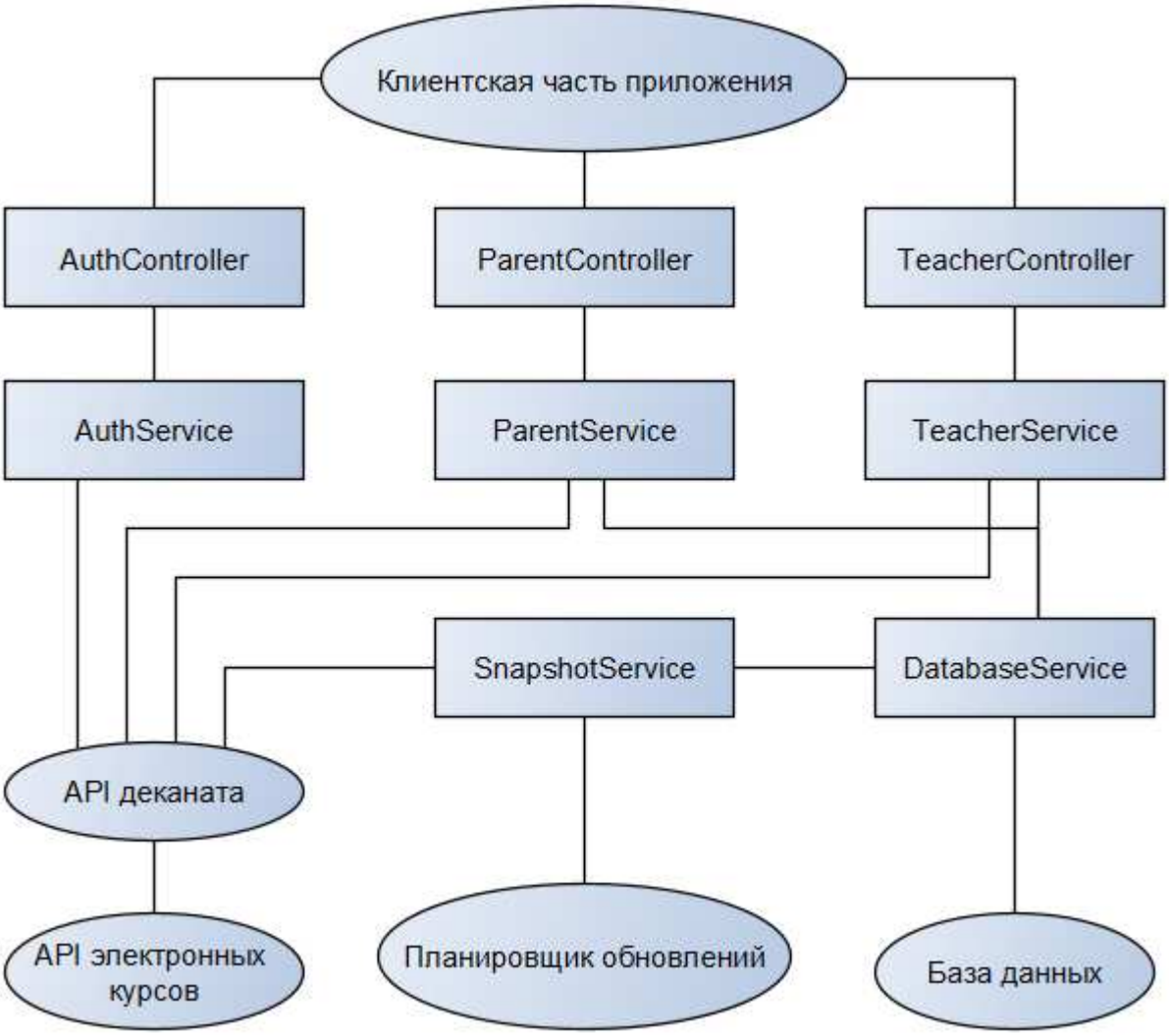


Рисунок 4 – архитектура системы

2.2 Архитектура базы данных

При проектировании архитектура базы данных требовалось опираться на уже существующую схему АСУ ИКИТ, представленную на рисунке 5. Высокая

связность таблиц и отсутствие доступа ко многим из них через API АСУ ИКИТ привели к необходимости прибегнуть к гораздо более простому устройству базы данных для приложения.

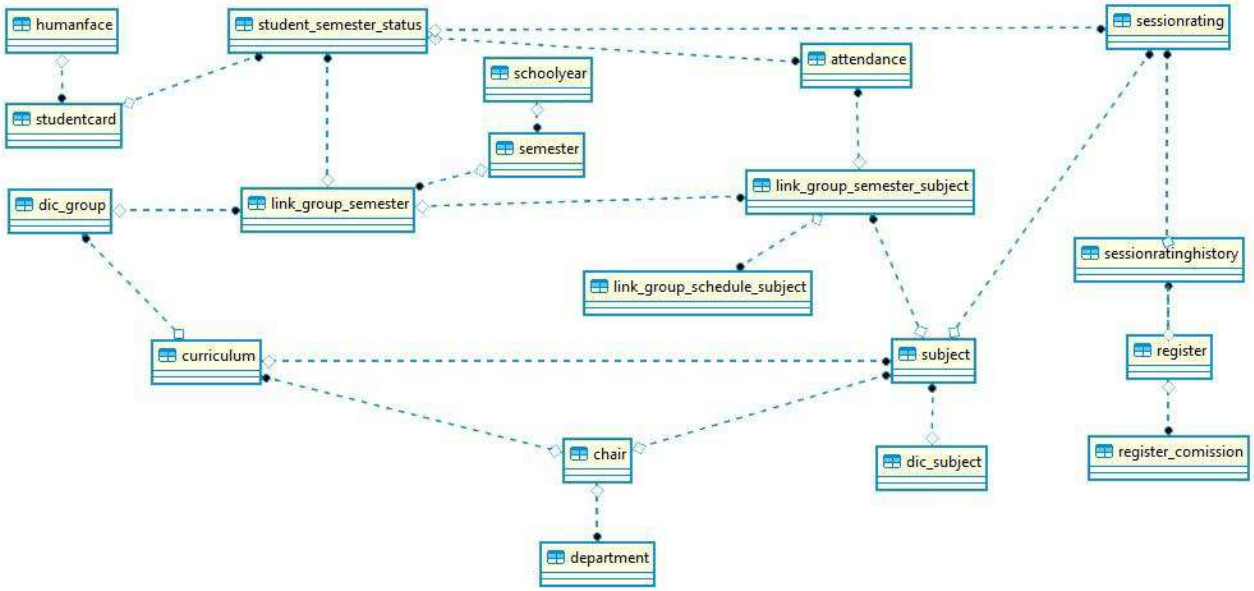


Рисунок 5 – схема данных АСУ ИКИТ

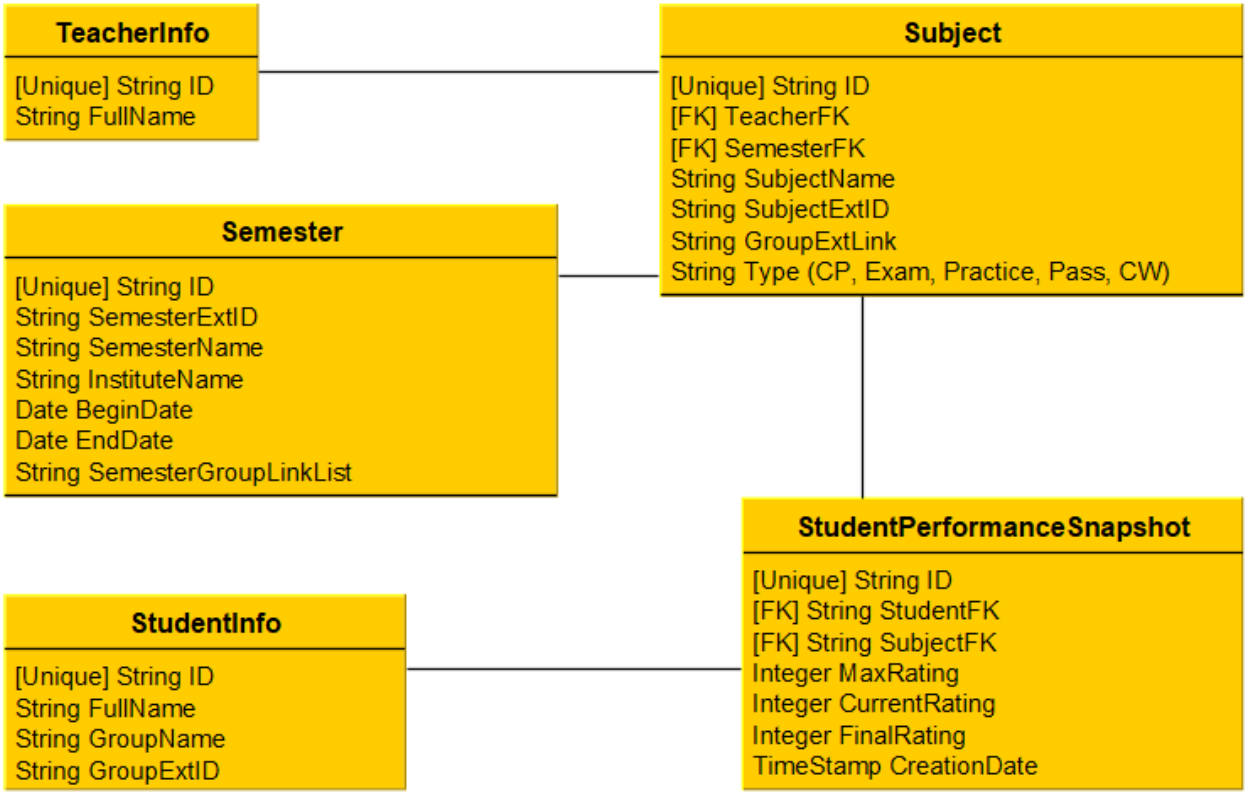


Рисунок 6 – схема данных приложения

Основная сущность, которую можно выделить в базе данных – снапшот успеваемости студента, представленный таблицей StudentPerformanceSnapshot. Снапшот составляется для каждого изучаемого студентом предмета.

Таблица 1 – StudentPerformanceSnapshot

Название поля	Тип данных	Ограничения	Описание
ID	Строка	Первичный ключ	Идентфикатор записи
StudentFK	Строка	Внешний ключ	Ссылка на студента
SubjectFK	Строка	Внешний ключ	Ссылка на предмет
MaxRating	Целое число		Максимально возможное количество баллов, которое можно получить по дисциплине в СЭО СФУ
CurrentRating	Целое число		Набранные баллы
FinalRating	Целое число		Оценка по дисциплине
CreationDate	Дата и время		Время формирования снапшота

TeacherInfo и StudentInfo – таблицы, позволяющие идентифицировать преподавателя и студента соответственно.

Таблица 2 – StudentInfo

Название поля	Тип данных	Ограничения	Описание
ID	Строка	Первичный ключ	Идентфикатор записи
FullName	Строка		ФИО студента
GroupName	Строка		Название учебной группы
GroupExtId	Строка		Идентификатор учебной группы в базе данных АСУ ИКИТ

Таблица 3 – TeacherInfo

Название поля	Тип данных	Ограничения	Описание
ID	Строка	Первичный ключ	Идентфикатор записи
FullName	Строка		ФИО преподавателя

Subject – сущность преподаваемой дисциплины, служит связью между снапшотом и преподавателем.

Таблица 4 – Subject

Название поля	Тип данных	Ограничения	Описание
ID	Строка	Первичный ключ	Идентификатор записи
TeacherFK	Строка	Внешний ключ	Ссылка на преподавателя
SemesterFK	Строка	Внешний ключ	Ссылка на семестр
SubjectName	Строка		Название дисциплины
SubjectExtID	Строка		Идентификатор дисциплины в базе данных АСУ ИКИТ
GroupExtLink	Строка		Идентификатор учебной группы в базе данных АСУ ИКИТ
Type	Строка	Возможные значения: CP, Exam, Practice, Pass, CW	Форма аттестации по предмету

Semester – таблица, которая используется для последующей фильтрации данных на стороне сервера.

Таблица 5 – Semester

Название поля	Тип данных	Ограничения	Описание
ID	Строка	Первичный ключ	Идентификатор записи

Окончание таблицы 5

Название поля	Тип данных	Ограничения	Описание
SemesterGroupLinkList	Строка		Список идентификаторов учебных групп, связанных с семестром
SemesterExtID	Строка		Идентификатор семестра в базе данных АСУ ИКИТ
SemesterName	Строка		Наименование семестра
InstituteName	Строка		Название института
BeginDate	Дата		Дата начала семестра
EndDate	Дата		Дата окончания семестра

2.3 Выводы по разделу

Целью раздела являлось описание архитектуры разрабатываемой back-end части приложения. Также была описана архитектура базы данных.

3 Разработка и тестирование

3.1 Разработка

Начальным этапом для разработки стало определение структуры файлов проекта, представленной на рисунке 7. Корневые разделы:

– controller, содержит классы контроллеров;

– dto, содержит классы моделей, определяющий структуру тела запросов и ответов от приложения;

– entity, содержит классы сущностей из базы данных;

– service, содержит классы сервисов;

– utils, содержит утилиты вроде класса-конфигуратора приложения;

– test.java, содержит классы тестов.

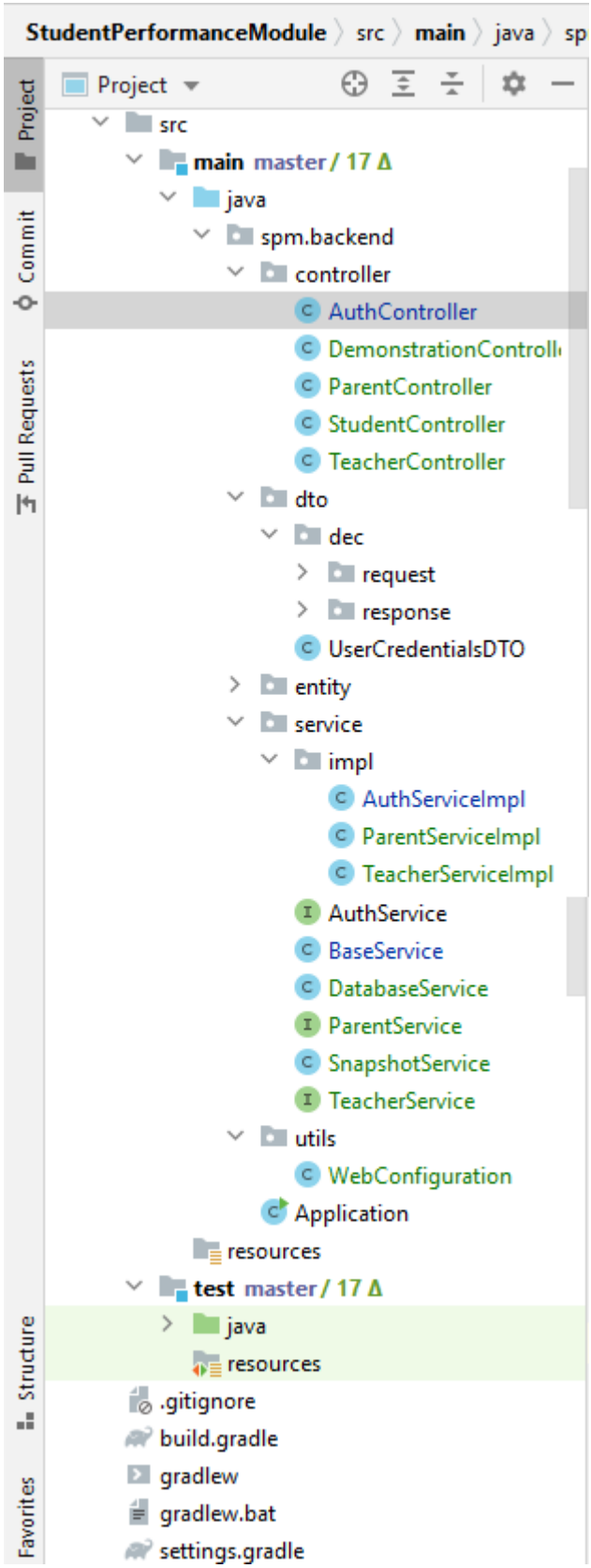


Рисунок 7 – фрагмент структуры проекта

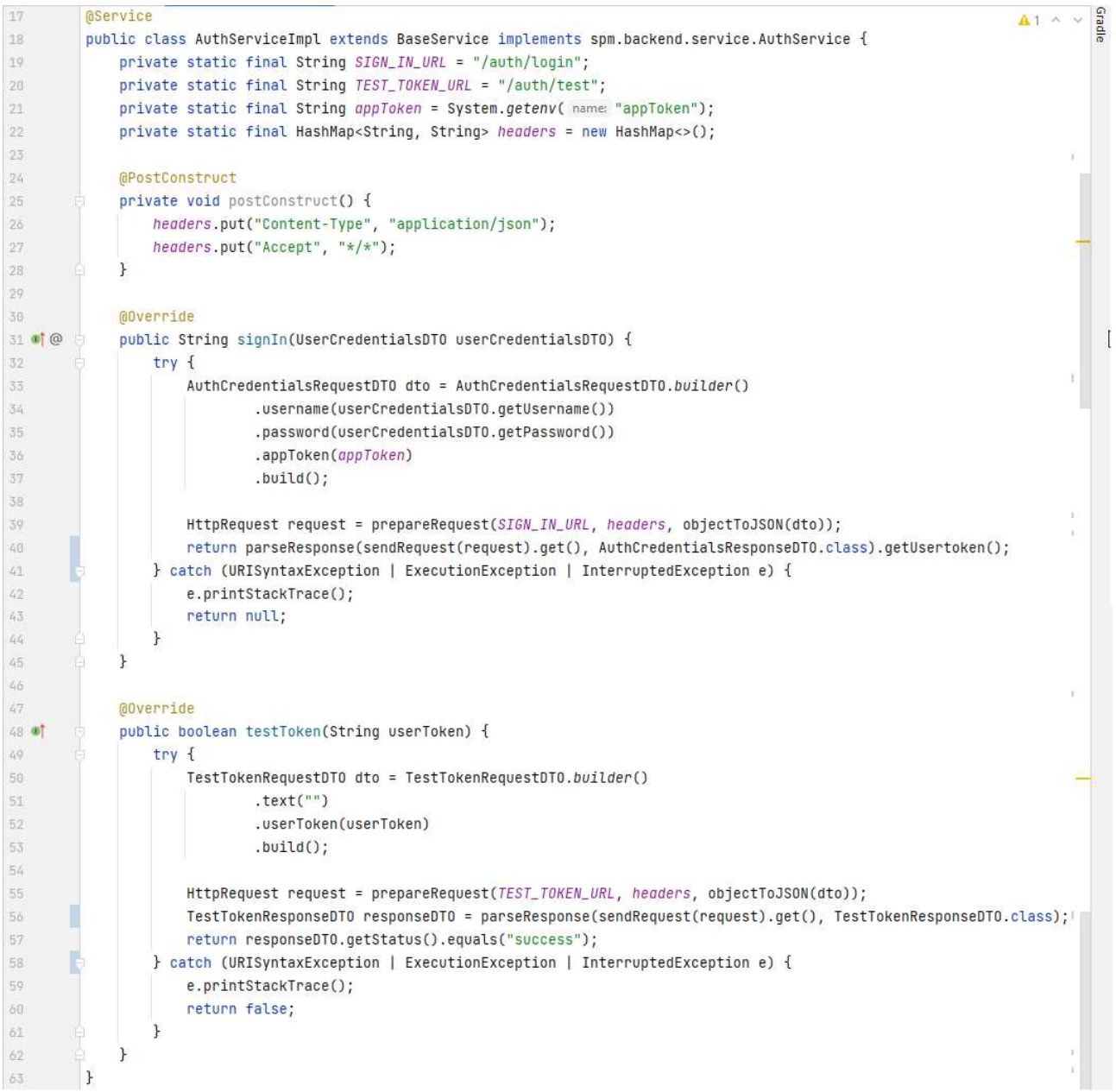
Классы контроллеров задают структуру API приложения с помощью аннотаций Spring. Например, AuthController с помощью аннотации @PostMapping даёт доступ к двум POST-методам по адресу *http://адрес_хоста/api/auth*: авто-

ризации (/sign-in) и проверки токена пользователя (/test-token). Аннотация @RequestBody позволяет задать структуру тела запроса, о чём будет рассказано ниже.

```
11  @RestController
12  @RequestMapping("api/auth")
13  public class AuthController {
14      private final AuthService service;
15
16      public AuthController(AuthServiceImpl service) { this.service = service; }
19
20      @PostMapping(value = "/sign-in",
21              consumes = "application/json")
22      public String signIn(@RequestBody UserCredentialsDTO userCredentialsDTO) {
23          return service.signIn(userCredentialsDTO);
24      }
25
26      @PostMapping(value = "/test-token")
27      public boolean testToken(@RequestBody String userToken) {
28          return service.testToken(userToken);
29      }
30  }  FlawlessTheory, 05.04.2021 0:26 · auth service
```

Рисунок 8 – AuthController

Как было сказано в разделе 2.1, считается хорошей практикой писать бизнес-логику в сервисах. AuthServiceImpl – пример подобного сервиса: например, при попытке авторизироваться будет вызван метод signIn(), представленный на рисунке 9.



```java
@Service
public class AuthServiceImpl extends BaseService implements spm.backend.service.AuthService {
    private static final String SIGN_IN_URL = "/auth/login";
    private static final String TEST_TOKEN_URL = "/auth/test";
    private static final String appToken = System.getenv( name: "appToken");
    private static final HashMap<String, String> headers = new HashMap<>();

    @PostConstruct
    private void postConstruct() {
        headers.put("Content-Type", "application/json");
        headers.put("Accept", "*/*");
    }

    @Override
    public String signIn(UserCredentialsDTO userCredentialsDTO) {
        try {
            AuthCredentialsRequestDTO dto = AuthCredentialsRequestDTO.builder()
                    .username(userCredentialsDTO.getUsername())
                    .password(userCredentialsDTO.getPassword())
                    .appToken(appToken)
                    .build();

            HttpRequest request = prepareRequest(SIGN_IN_URL, headers, objectToJSON(dto));
            return parseResponse(sendRequest(request).get(), AuthCredentialsResponseDTO.class).getUsertoken();
        } catch (URISyntaxException | ExecutionException | InterruptedException e) {
            e.printStackTrace();
            return null;
        }
    }

    @Override
    public boolean testToken(String userToken) {
        try {
            TestTokenRequestDTO dto = TestTokenRequestDTO.builder()
                    .text("")
                    .userToken(userToken)
                    .build();

            HttpRequest request = prepareRequest(TEST_TOKEN_URL, headers, objectToJSON(dto));
            TestTokenResponseDTO responseDTO = parseResponse(sendRequest(request).get(), TestTokenResponseDTO.class);
            return responseDTO.getStatus().equals("success");
        } catch (URISyntaxException | ExecutionException | InterruptedException e) {
            e.printStackTrace();
            return false;
        }
    }
}
```

Рисунок 9 - AuthServiceImpl

Следует отметить, что методы для работы с API АСУ ИКИТ определены в базовом классе BaseService.

Рисунок 10 – BaseService

Как было сказано выше, указание модели как аргумента функции контроллера с аннотацией @RequestBody позволяют задать формат тела запроса для контроллера

Рисунок 11 – UserCredentialsDTO

Вся логика по работе с данными заключена в сервисе DatabaseService. Пример по получению записи о студенте показан на рисунке 12.

```java
@Service
public class DatabaseService {
    public Student getStudent(String ID) throws Exception {
        String url = "jdbc:postgresql://localhost:5432/testdb";
        String user = "user12";
        String password = "34klq*";

        String query = "SELECT * FROM StudentInfo WHERE ID = 1";
        Student student = new Student();

        try (Connection con = DriverManager.getConnection(url, user, password);
             PreparedStatement pst = con.prepareStatement(query)) {
            pst.setString( parameterIndex: 1, ID);
            boolean isResult = pst.execute();

            do {
                try (ResultSet rs = pst.getResultSet()) {

                    while (rs.next()) {
                        student.setID(rs.getString( columnIndex: 1));
                        student.setFullName(rs.getString( columnIndex: 2));
                        student.setGroupId(rs.getString( columnIndex: 3));
                        student.setGroupExtId(rs.getString( columnIndex: 4));
                    }

                    isResult = pst.getMoreResults();
                }
            } while (isResult);

        } catch (SQLException ex) {
            throw new Exception(ex);
        }

        return student;
    }
```

Рисунок 12 – метод getStudent()

Для создания таблиц в базе данных использовалась утилита pgAdmin 4.

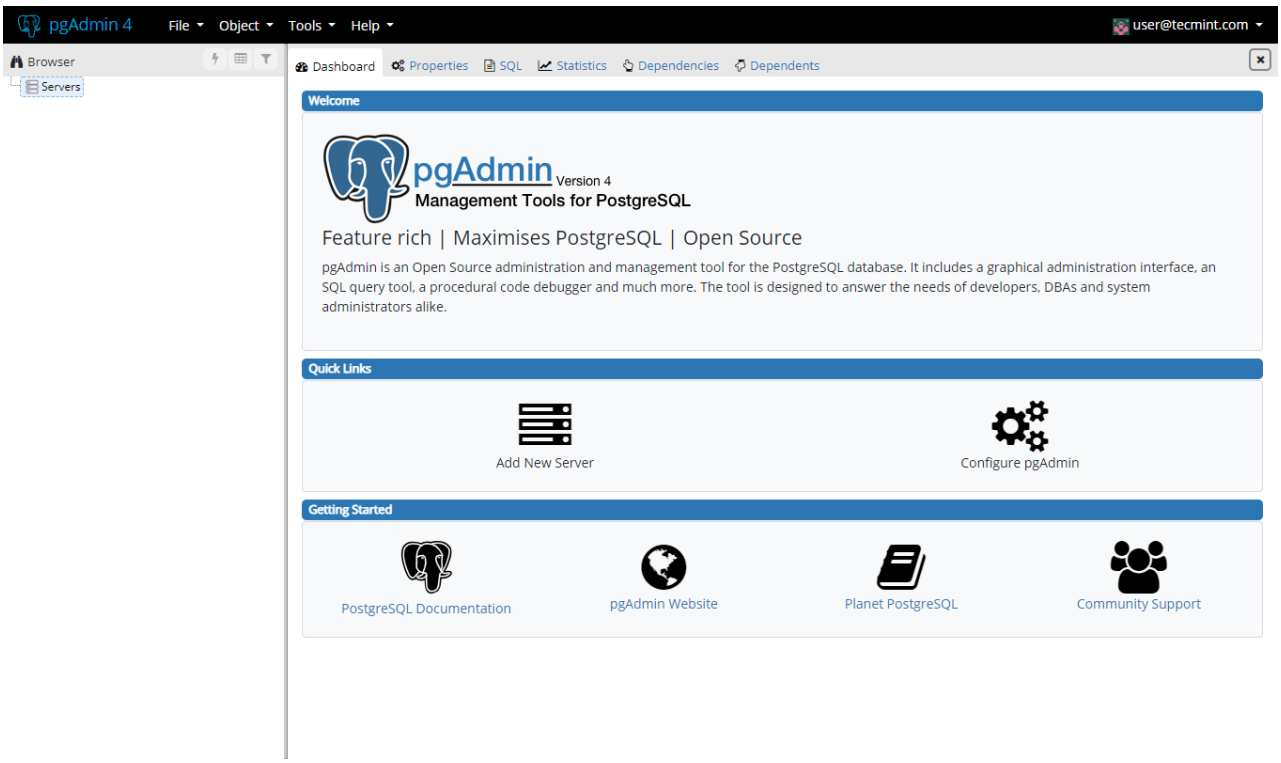


Рисунок 13 – интерфейс pgAdmin 4

При реализации планировщика обновлений была встречена проблема: так как все методы API АСУ ИКИТ завязаны на отдельно взятого пользователя, приложение не может проводить массовые операции вроде формирования снапшотов успеваемости: для этого потребовалось бы авторизироваться под учётными записями всех студентов – для получения их оценок, и преподавателей – для записи в базу данных сведений о преподаваемых ими дисциплинах. Решение было найдено в доработке API и добавлении в него учётной записи «суперпользователя», который может получить данные по любой другой учётной записи. Стоит отметить, что серьёзных угроз для безопасности АСУ ИКИТ это нововведение не создало, так как, несмотря на название, «суперпользователь» может лишь просматривать произвольные данные, но не редактировать их и не добавлять новые.

Для того чтобы планировщик обновлений работал, необходимо включить выполнение задач по расписанию в конфигурации приложения. Так как в данном случае для конфигурирования используется специальный класс WebConfiguration, это делается с помощью добавления аннотации @EnableScheduling.

Планировщик обновлений – это специальный класс, аннотированный как @Component. Его главное отличие от остальных классов – метод execute(), последовательно вызывающий шаги обновления. Аннотация метода @Scheduled(cron = «0 0 0 ? * SUN *») означает, что он будет выполняться в полночь каждого воскресенья.

Алгоритм обновления:

– приложение авторизируется в АСУ ИКИТ как «суперпользователь»;

– запрашивается информации о преподавателях, проводится её обработка;

– если были найдены новые записи, они добавляются в таблицу TeacherInfo;

– для каждого из преподавателей запрашивается список преподаваемых дисциплины;

– если были найдены отсутствующие в базе данных дисциплины, они добавляются в таблицу Subject;

– запрашивается информация о студентах, проводится её обработка;

– если были найдены новые записи, они добавляются в таблицу StudentInfo;

– у СЭО СФУ запрашиваются оценки каждого студента;

– на основе этих оценок формируется снапшот успеваемости;

– снапшот связывается с дисциплиной и записывается в базу данных.

3.2 Тестирование

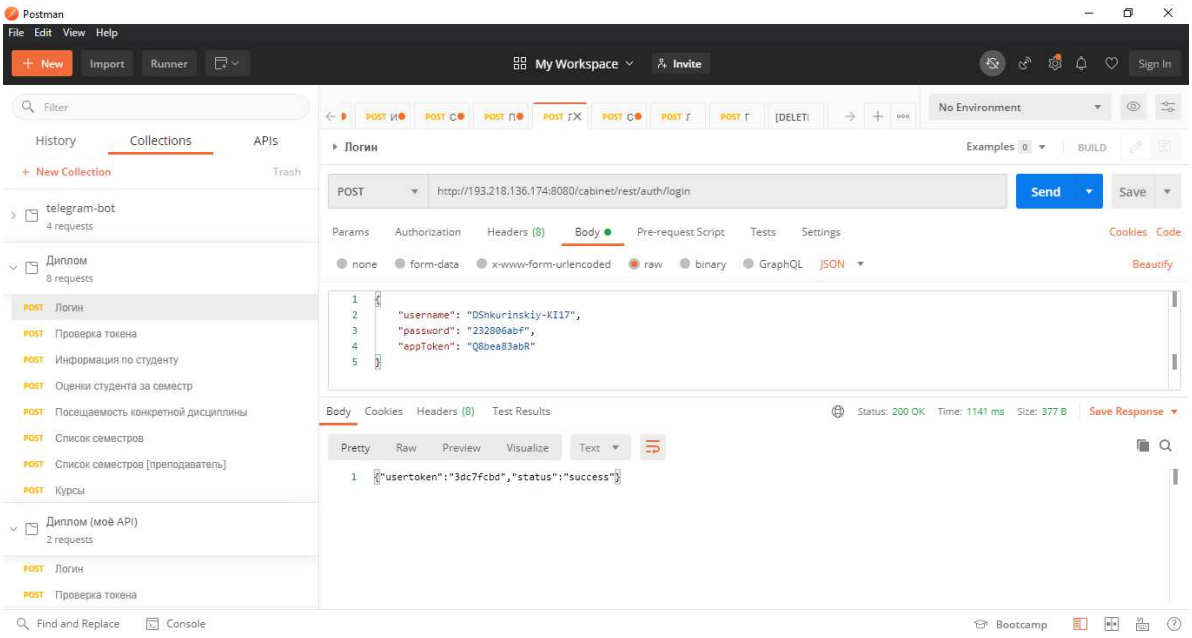


Рисунок 14 – интерфейс Postman

В ходе тестирования механизма формирования слепков была встречена проблема: превышался лимит на количество запросов к СЭО СФУ. Попытки решить проблему путём переговоров о повышении лимита с ответственными лицами не принесли результата. Переработку механизма не удалось закончить в установленный срок, из-за чего было решено отложить данный вопрос на потом.

3.3 Выводы по разделу

Целью раздела являлось описание ключевых моментов разработки и тестирования.

В разделе 3.1 была описана работа с фреймворком Spring Boot Starter Web и библиотеками Jackson и Lombok, СУБД PostgreSQL и утилитой pgAdmin 4. В разделе 3.2 была описана работа с ПО Postman для тестирования приложения.

4 Описание результатов разработки

4.1 Back-end

Результатом разработки стало серверное приложение с картой API, представленной в таблицах 6-8.

Таблица 6 – карта API (авторизация)

Путь	Метод	Описание	Входные данные	Выходные данные

/sign-in	POST	Авторизация (преподаватель)	{
			login: «login»,
			password: «pass»,
			appToken: «123»
			}
				{
				ID: «ID записи в базе данных»,
				firstName: «имя»,
				lastName: «фамилия»,
				patronymic: «отчество»,
				type: «teacher»,
				token: «токен пользователя»,
				subjects: [
				{
				ID: «ID дисциплины»,
				subjectName: «название дисциплины»
				},
				…
				]
				}

Окончание таблицы 6

Путь	Метод	Описание	Входные данные	Выходные данные

/sign-in	POST	Авторизация (родитель)	{
			login: «login»,
			password: «pass»,
			appToken: «123»
			}
				{
				ID: «ID записи в базе данных»,
				firstName: «имя родителя»,
				lastName: «фамилия родителя»,
				patronymic: «отчество родителя»,
				type: «parent»,
				token: «токен пользователя»,
				student: {
				ID: «ID записи в базе данных»,
				firstName: «имя студента»,
				lastName: «фамилия студента»,
				patronymic: «отчество студента»,
				semesterList: [
				ID: «ID семестра»,
				semesterName: «название семестра»
				]
				}
				}

/test-token	POST	Проверка токена	Токен пользователя	«true» или «false», если срок действия токена истёк

Таблица 7 – карта API (родитель)

Путь	Метод	Описание	Входные данные	Выходные данные

/parent/performance	POST	Возвращает информацию об успеваемости студента в выбранном семестре, включая позицию в рейтинге успеваемости в группе (высчитывается на основе успеваемости в курсах)	{
			semesterID: «ID семестра»
			token: «токен пользователя»
			}
				{
				ratingPosition: «X из Y»,
				totalPerformance: процент усвоения учебной программы,
				attendance: процент посещаемости,
				subjectInfo: [
				{
				ID: «ID дисциплины»,
				SubjectName: «название дисциплины»,
				completion: процент выполнения курса,
				controlKind: «форма контроля»,
				controlMark: оценка за предмет,
				date: дата прохождения экзамена/зачёта
				},
				…
				]
				}

Таблица 8 – карта API (преподаватель)

Путь	Метод	Описание	Входные данные	Выходные данные

/teacher/subjectDates	POST	Возвращает массив лет, в которые преподавалась данная дисциплина	{
			subjectID: «ID дисциплины»,
			token: «токен пользователя»
			}
				Массив целых чисел

/teacher/subjectGroups	POST	Возвращает список групп, которым в указанные года преподавалась данная дисциплина	{
			subjectID: «ID дисциплины»,
			years: [массив целых чисел],
			token: «токен пользователя»
			}
				{
				groups: [
				{
				ID: «идентификатор группы»,
				groupName: «название группы»
				},
				…
				]
				}

Окончание таблицы 8

Путь	Метод	Описание	Входные данные	Выходные данные

/teacher/performanceReport	POST	Возвращает объект отчёта по успеваемости групп	{
			subjectID: «ID дисциплины»,
			years: [массив целых чисел],
			groups: [
			{
			ID: «идентификатор группы»,
			groupName: «название группы»
			},
			…
			],
			token: «токен пользователя»
			}
				см. рисунок 15

/teacher/aggregatedPerformanceReport	POST	Возвращает объект агрегированного отчёта по успеваемости групп	{
			subjectID: «ID дисциплины»,
			years: [массив целых чисел],
			groups: [
			{
			ID: «идентификатор группы»,
			groupName: «название группы»
			},
			…
			],
			token: «токен пользователя»
			}
				см. рисунок 16

```
reports: [
    {
        groupName: "Название группы",
        groupID: "ID группы",
        students: количество студентов в группе,
        passed: количество студентов, получивших оценку по предмету,
        courseStat: [
            {
                progress: '0-9',
                count: кол-во студентов, освоивших дисциплину в пределах от 0 до 9 процентов
            },
            ... // ещё 5 объектов с шагом в 20 процентов (10-20, 20-40, 40-60 и т. д.), 7 объект - progress: "100"
        ],
        attendanceLecture: [
            {
                ID - "Макс."
                data: [
                    {
                        x: "Название первого месяца в семестре",
                        y: посещаемость
                    },
                    ...
                ]
            },
            ... // Ещё два объекта: "Мин." и "Сред."
        ],
        attendancePractice: [
            {
                ID - "Макс."
                data: [
                    {
                        x: "Название первого месяца в семестре",
                        y: средняя посещаемость,
                        max: макс. посещаемость,
                        min: мин. посещаемость
                    },
                    ...
                ]
            },
            ... // Ещё два объекта: "Мин." и "Сред."
        ],
        marks: [
            {
                id: "на 5"/"зачёт",
                label: "на 5"/"зачёт",
                value: кол-во студентов, сдавших на указанную оценку
            },
            ...
        ],
        debts: [
            {
              id: "1 долг",
              label: "1 долг",
              value: кол-во студентов, имеющих данное количество задолженностей
            },
            ...
        ]
    },
    ... // Повторяется до "Больше 3 долгов"
]
```

Рисунок 15 – тело отчёта успеваемости групп

```
{
    groups: [
        {
            ID: "Идентификатор группы",
            groupName: "Название группы"
        },
        ...
    ],
    students: количество студентов в группах,
    passed: количество студентов, получивших оценку по дисциплине,
    courseStat: [
        {
            progress: "0-9",
            count: кол-во студентов, освоивших дисциплину в пределах от 0 до 9 процентов
        },
        ... // ещё 5 объектов с шагом в 20 процентов (10-20, 20-40, 40-60 и т. д.), 7 объект - progress: "100"
    ],
    attendanceLecture: [
       {
          ID - "Название группы",
          actualID - "ID группы",
          data: [
            {
              x: "Название первого месяца в семестре",
              y: средняя посещаемость среди всех групп
            },
            ...
          ]
       },
       ...
    ],
    attendancePractice: [
       {
          ID - "Название группы",
          actualID - "ID группы",
          data: [
            {
              x: "Название первого месяца в семестре",
              y: средняя посещаемость среди всех групп
              max: "Макс. посещаемость",
              min: "Мин. посещаемость"
            },
            ...
          ]
       },
       ...
    ],
    marks: [
        {
            id: "на 5"/"зачёт",
            label: "на 5"/"зачёт",
            value: кол-во студентов, сдавших на указанную оценку
        },
        ...
    ],
    debts: [
```

Рисунок 16 – тело агрегированного отчёта

4.2 Выводы по разделу

Целью данного раздела являлось описание результатов разработки веб-приложения.

Была представлена схема API приложения.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы был проведён анализ существующих решений, были составлены требования, описан выбор стека разработки, реализованы этапы проектирования и разработки серверной части.

Результатом выпускной квалификационной работы стало веб-приложение для просмотра информации о студентах на основе СЭО СФУ.

Разработанное приложение позволяет:

– авторизоваться;

– просматривать учебную информацию ребенка (для родителей);

– просматривать агрегированный отчёт по выбранным группам (для преподавателей);

– просматривать отчёт по каждой выбранной группе отдельно (для преподавателей).

Были выполнены все задачи, кроме реализации функционала для преподавателей. Для дальнейшего развития системы первостепенной задачей является обход ограничения на максимальное количество запросов. Также имеются иные способы улучшения системы, более частое формирование снапшотов, реализация дополнительных фильтров для преподавателей и разработка модуля для работников деканата.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Введение в REST API - RESTful веб-сервисы / Хабр [Электронный ресурс] // Портал Хабр. Режим доступа: https://habr.com/ru/post/483202/.

2. Жизненный цикл разработки ПО. [Электронный ресурс] // Сайт компании XB Software. – Режим доступа: https://xbsoftware.ru/blog/zhiznennyj-tsykl-po-kanban/.

3. Многоуровневая архитектура [Электронный ресурс] // Сайт сибирского отделения Российской академии наук. – Режим доступа: http://wwwsbras.nsc.ru/Report2006/Report321/node30.html.

4. Общие архитектуры веб-приложений [Электронный ресурс] // Официальный сайт Microsoft. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures.

5. Паттерны для новичков: MVC vs MVP vs MVVM / Хабр [Электронный ресурс] // Портал Хабр. Режим доступа: https://habr.com/ru/post/215605/.

6. Что такое MVC: рассказываем простыми словами [Электронный ресурс] // Блог компании Hexlet. Режим доступа: https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami.

7. Cache для анализа неструктурированных данных | InterSystems [Электронный ресурс] // Сайт компании InterSystems. Режим доступа: https://www.intersystems.com/ru/products/cache/.

8. Connect to PostgreSQL with JDBC driver - Mkyong.org [Электронный ресурс] // Портал Mkyong. Режим доступа: https://mkyong.com/jdbc/how-do-connect-to-postgresql-with-jdbc-driver-java/.

9. Getting Started | Serving Web Content with Spring MVC [Электронный ресурс] // Официальный сайт Spring. Режим доступа: https://spring.io/guides/gs/serving-web-content/.

10. GitHub - FasterXML/jackson: Main portal page for the Jackson project [Электронный ресурс] // Репозиторий GitHub. Режим доступа: https://github.com/FasterXML/jackson.

11. IntelliJ IDEA: функциональная и эргономичная IDE для разработки на Java от JetBrains [Электронный ресурс] // Официальный сайт компании JetBrains. Режим доступа: https://www.jetbrains.com/ru-ru/idea/.

12. Introducing ObjectScript - Using Cache ObjectScript - Cache & Ensemble 2018.1.5 [Электронный ресурс] // Портал документации InterSystems. Режим доступа: https://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=GCOS_INTRO.

13. JUnit 5 [Электронный ресурс] // Официальный сайт проекта JUnit. Режим доступа: https://junit.org/junit5/.

14. Modern Student Informaion System Software | Cloud SIS Software [Электронный ресурс] // Сайт организации Creatrix Campus. Режим доступа: https://www.creatrixcampus.com/student-information-system.

15. PostgreSQL: The world's most advanced open source database [Электронный ресурс] // Официальный сайт PostgreSQL. Режим доступа: https://www.postgresql.org.

16. PostgreSQL Java - PostgreSQL programming in Java [Электронный ресурс] // Сайт ZetCode. Режим доступа: https://zetcode.com/java/postgresql/.

17. Postman | The Collaboration Platform for API Development [Электронный ресурс] // Официальный сайт Postman. Режим доступа: https://www.postman.com.

18. Project Lombok [Электронный ресурс] // Официальный сайт проекта Lombok. Режим доступа: https://projectlombok.org.

19. Student Performance Tracking [Электронный ресурс] // Сайт организации eduCloud Infotech. Режим доступа: https://www.educloud.in/lms/student-performance.jsp.

20. Teaching | Engage School MIS [Электронный ресурс] // Сайт организации Engage. Режим доступа: https://engagesmis.com/engage-teaching-mis/.

21. TortoiseGit - Windows Shell Interface to Git [Электронный ресурс] // Официальный сайт TortoiseGit. Режим доступа: https://tortoisegit.org.

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра «Информатика»

УТВЕРЖДАЮ
Заведующий кафедрой

_____ А. С. Кузнецов
подпись инициалы, фамилия

«17» 06 2021 г

БАКАЛАВРСКАЯ РАБОТА

09.03.04 Программная инженерия

Разработка back-end части модуля анализа успеваемости студентов на основе СЭО СФУ

Руководитель ВКР 17.06.21. доцент, канд. техн. наук А. В. Хныкин
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник 17.06.21 Д. О. Шкуринский
подпись, дата инициалы, фамилия

Красноярск 2021