ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО
Приглашенный преподаватель факультета компьютерных наук департамента программной инженерии

Г. М. Сосновский
«08» мая 2022 г.

УТВЕРЖДАЮ
Академический руководитель образовательной программы «Программная инженерия»

В. В. Шилов
«08» мая 2022 г.

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
RU.17701729.04.05 -01 81				

«HSE APPLE» – ОБРАЗОВАТЕЛЬНАЯ СОЦИАЛЬНАЯ СЕТЬ ДЛЯ НИСа ПО iOS РАЗРАБОТКЕ

Пояснительная записка

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.04.05-01 81 01-1-ЛУ

Исполнитель:
студент группы БПИ204

/ Д. А. Тасбауова /
«08» мая 2022 г.

Москва 2022

УТВЕРЖДЕН
RU.17701729.04.05-01 81 01-1-ЛУ

Пояснительная записка

RU.17701729.04.05-01 81 01-1

Листов 83

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
RU.17701729.04.05 -01 81				

Москва 2022

СОДЕРЖАНИЕ

АННОТАЦИЯ

В данном программном документе приведена пояснительная записка к программе ««HSE Apple» образовательная социальная сеть для НИСа по iOS разработке»

В разделе «Введение» указано наименование программы, краткое наименование программы, документы, на основании которых ведется разработка, а также организация, утвердившая данный документ.

В разделе «Назначение и область применения» указано функциональное назначение программы, эксплуатационное назначение программы и краткая характеристика области применения программы.

В разделе «Технические характеристики» содержатся следующие подразделы:

1) постановка задачи на разработку программы;

2) описание алгоритма и функционирования программы с обоснованием выбора схемы алгоритма решения задачи и возможные взаимодействия программы с другими программами;

3) описание и обоснование выбора метода организации входных и выходных данных;

4) описание и обоснование выбора состава технических и программных средств.

В разделе «Ожидаемые технико-экономические показатели» указана предполагаемая потребность и экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами.

Настоящий документ разработан в соответствии с требованиями:

1) ГОСТ 19.101–77 Виды программ и программных документов [1];

2) ГОСТ 19.102–77 Стадии разработки [2];

3) ГОСТ 19.103–77 Обозначения программ и программных документов [3];

4) ГОСТ 19.104–78 Основные надписи [4];

5) ГОСТ 19.105–78 Общие требования к программным документам [5];

6) ГОСТ 19.106–78 Требования к программным документам, выполненным печатным способом [6];

7) ГОСТ 19.404–79 Пояснительная записка. Требования к содержанию и оформлению [7].

Изменения к Пояснительной записке оформляются согласно ГОСТ 19.603–78 [8], ГОСТ 19.604–78 [9].

Перед прочтением данного документа рекомендуется ознакомиться с терминологией, приведенной в Приложении 1.

1. ВВЕДЕНИЕ

1.1. Наименование программы

Наименование: ««HSE Apple» – образовательная социальная сеть для НИСа по iOS разработке».

Наименование на английском языке: ««HSE Apple» – educational social network for a research seminar on iOS development».

1.2. Документы, на основании которых ведется разработка

Основанием для разработки является учебный план подготовки бакалавров по направлению 09.03.04 "Программная инженерия" и утвержденная академическим руководителем тема курсового проекта.

2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

2.1. Назначение программы

2.1.1. Функциональное назначение

Основными целями создания приложения являются автоматизация информационного обмена, эффективное управление и создание единого информационного пространства, предоставление необходимой информации заинтересованным пользователям приложения.

Основные функции приложения:

1. возможность коммуникации между пользователями приложения
2. возможность прохождение контроля уровня знаний (тесты, лабораторные работы)
3. возможность размещения учебных материалов (лабораторные задания, тесты, медиаконтент, ссылки на внешние источники и т.д.), данных учебного процесса (события, новости, объявления, расписание и т.п.) и данных контроля обучения (дедлайны, ведомость с оценками).

Функциональным назначением серверной составляющей проекта является обработка запросов, предоставление данных для клиентской части и хранение данных приложения.

Функции серверной части:

1. Прием запросов от приложений-клиентов.
2. Интерпретация запросов.
3. Оптимизация и выполнение запросов к БД.
4. Отправка результатов приложению-клиенту.
5. Обеспечение системы безопасности и разграничение доступа.
6. Управление целостностью БД.

2.1.2. Эксплуатационное назначение

Для достижения стоящих перед системой целей Backend-подсистема должна обеспечить выполнение следующих основных задач:

1. ввода, валидации, обработки, систематизации, хранения, восстановления, контроля и получения информации;

2. формирование и работа с базой данных;

3. организация решений по обеспечению безопасности;

4. обеспечение надежности функционирования системы.

Конечным пользователем является клиентская часть приложения (Android, iOS, Web).

В целях безопасности права пользователей на функционал в рамках приложения строго ограничены в соответствии с ролевой моделью доступа.

Роль пользователя	Права на функционал в приложении
Преподаватель	Создание и изменение: группы для общения, контрольные задания, ведомости, объявления. Присвоение и изменение прав для пользователей.
Ассистент	Имеет разрешение на редактирование ведомости, проверки контрольных заданий студентов.
Студент	Имеет доступ к группам, заданиям, только своим оценкам, обьявлениям. Но не может редактировать их.

2.2. Краткая характеристика области применения

Краткая характеристика области применения: «“HSE Apple” – серверная часть образовательной социальной сети для проведения НИСа по iOS-разработке» - серверная часть приложения, предназначающегося для курса НИСа среди студентов «НИУ ВШЭ»

3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

3.1. Постановка задачи на разработку программы

Программа должна обеспечивать возможность выполнения следующих функций:

Функции разделены на несколько блоков, имеющих общий логический контекст:

1) Контроль сеанса работы пользователя
2) Профиль аккаунта и пользовательские настройки приложения
3) Навигация и лента событий
4) Чат и посты
5) Задания (лабораторные работы)
6) Ведомости успеваемости

1. Функционал для Контроля аккаунтов и сеанса работы аккаунтов
1.1. Авторизация аккаунта
2. Функционал Профиля аккаунта и пользовательских настроек приложения
2.1. Добавление/изменение данных профиля
2.2. Предоставление данных профиля
3. Функционал Навигации и ленты событий.
3.1. Добавление/изменение курсов
3.2. Добавление/изменение событий
3.3. Предоставление данных события/списка событий
4. Функционал Чата и постов.
4.1. Удаление чата
4.2. Предоставление данных чата/списка чатов
4.3. Добавление/изменение/удаление поста
4.4. Предоставление данных поста/списка постов
5. Функционал Заданий
5.1. Добавление/изменение/удаление заданий
5.2. Предоставление данных задания/списка заданий
5.3. Добавление/изменение/удаление выполненных заданий

5.4. Предоставление данных выполненного задания/списка выполненных заданий

6. Функционал Ведомостей успеваемости

6.1. Добавление/изменение/удаление оценок

6.2. Предоставление данных ведомости/списка ведомостей

3.2. Описание алгоритма и функционирования программы

3.2.1. Описание построения серверной части

В проекте была выбрана архитектура MVC(Model-View-Controller), которая разделена на три основных компонента, отвечающих за решение различных задач:

1) Модель – этот компонент отвечает за данные и методы работы с ними. Это базы данных и фреймворк Object/Relational Mapping (ORM), Hibernate предоставляющий доступ к БД.

2) Контроллер – этот компонент обрабатывает все запросы пользователя и связывает модель и представления клиента.

3) Представление – это пользовательский интерфейс и представление клиента.

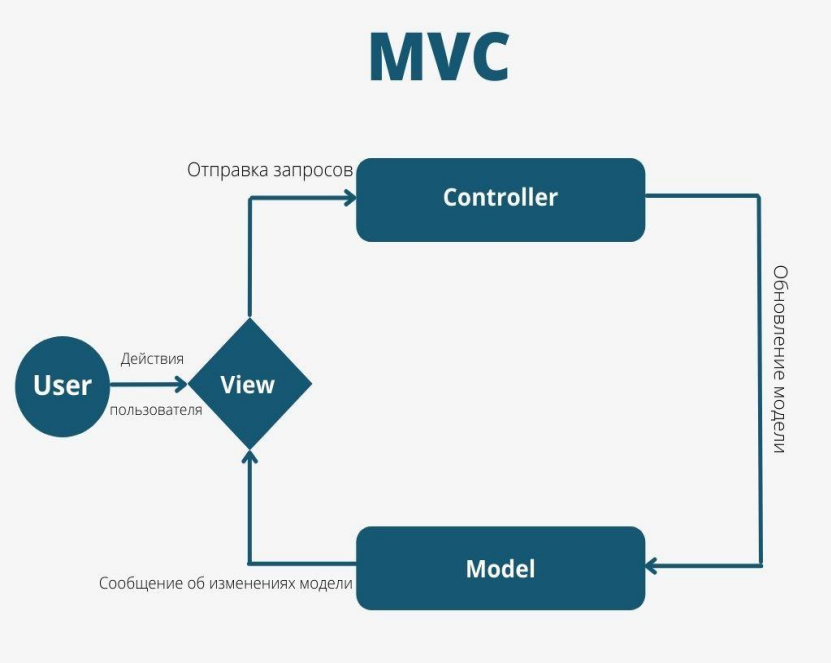

Рисунок 1 - архитектура MVC

3.2.1.1. Структура серверной части

Структура серверной части разделяется на 3 слоя:

1) Слой Controllers: содержат логику приложения и передают вводимые пользователем данные сервису;

2) Слой Service: промежуточное звено между контроллером и хранилищем. Получают данные от контроллера, выполняют валидацию и бизнес-логику, а также вызывают репозитории для обновления данных;

3) Слой Repository: слой для взаимодействия с моделями и выполнения операций с БД.

Слои и их взаимодействие представлены на рисунке 2.

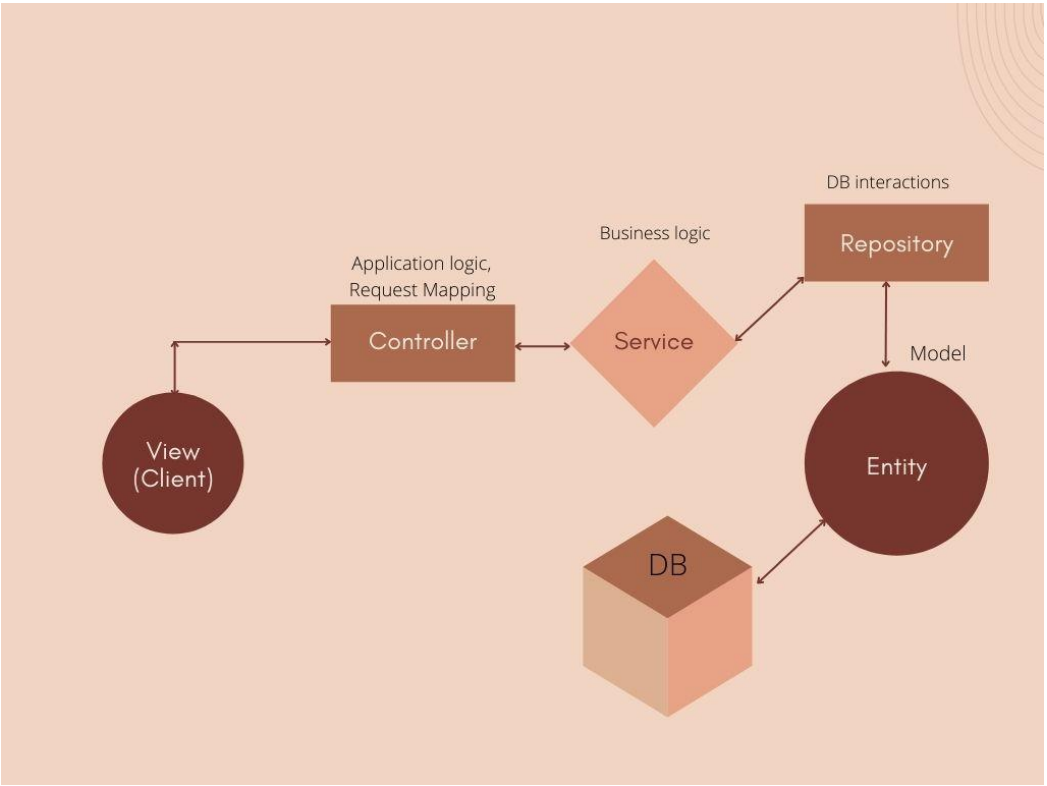

Рисунок 2 - слои серверной части

Контроллеры

В проекте используются 5 контроллеров, для управления и обработки HTTP запросов и обеспечения связи между пользователем и сервером.

1. UserController — контроллер, отвечающий за информацию о пользователе.
2. CourseController — контроллер, отвечающий за информацию о курсах.
3. TaskController - контроллер, отвечающий за управление информации о заданиях.

4. PostController - контроллер, отвечающий за управление информации о постах/новостях.
5. ChatController - контроллер, отвечающий за чаты, сообщения пользователей

```
@RestController
@SecurityRequirement(name = "Authorization")
@Tag(description = "Api to manage users",
      name = "User Resource")
public class UserController {

   @Autowired
   UserService userService;

   Logger logger = LoggerFactory.getLogger(UserController.class);

   @Operation(summary = "Get user for course",
         description = "Provides user for course by id")
   @RequestMapping(value = "user/{userID}/application/approved", method =
RequestMethod.GET)
   @ResponseBody
   @PreAuthorize("hasAnyAuthority('TEACHER', 'STUDENT', 'ASSIST')")
   public UserEntity getUser(@PathVariable("userID") Long userID){
      return userService.findUser(userID);
   }


   @Operation(summary = "Update user",
         description = "Provides new updated user.Access role - TEACHER, STUDENT,
ASSIST")
   @PreAuthorize("hasAnyAuthority('TEACHER', 'STUDENT', 'ASSIST')")
   @RequestMapping (value = "/user/{userID}", method = RequestMethod.PUT)
   public UserEntity updateUser(@RequestBody UserEntity newUser, @PathVariable("userID")
Long userID) {
      return userService.changeUser(userID, newUser);
   }
```

Листинг 1 - язык Java, UserController

Сервисы

В проекте используются 5 сервисов, соответственно: UserService, CourseService, TaskService, PostService, ChatService, в которых прописана вся бизнес-логика, которая в зависимости от действий пользователя, подключается к репозиторию.

```
@Service
    public class UserService {
        private static final Integer ROLE_COUNT = 3;

        @Autowired
        UserDao userDao;

        @Autowired
        CourseDao courseDao;

        public UserEntity findUser(Long userID) throws UsernameNotFoundException {
            return userDao.findById(userID).orElseThrow(() -> new
    BusinessException(ExceptionMessage.user_not_found));
        }

        public UserEntity changeUser(Long userID, UserEntity newUser) throws
    UsernameNotFoundException{
            UserEntity user = userDao.findById(userID).orElseThrow(() -> new
    BusinessException(ExceptionMessage.user_not_found));
            user.setCommonname(newUser.getCommonname());
            user.setEmail(newUser.getEmail());
            userDao.save(user);
            return user;
        }
```

Листинг 2 - язык Java, UserService

Также некоторая логика была вынесена в отдельные пакеты:

1) Пакет security: JWTAuthFilter, SecurityConfig, UserAndRole

В связи с тем, что аутентификация пользователя происходит на внешнем сервере (HSE Auth), с которым взаимодействует только клиентская часть, в серверной части происходит только авторизация. Каждый запрос требует передачи access токена, по которому идентифицируется пользователь. Токен передаётся в Header с ключом «Authorization».

JWTAuthFilter - класс, который декодирует полученный JWT Token с помощью библиотеки "com.auth0:java-jwt" и получает информацию о пользователе. Сохраняет информацию о его ролях, email, ФИО в класс UserAndRole, который реализует интерфейс UserDetails.

SecurityConfig - класс, который автоматически обеспечивает защиту для указанных там HTTP путей. Указаны URL запросов с ограниченным доступом, также ограничение задано, в зависимости от ролей.

2) Пакет error: AppException, BusinessException, TechnicalException, ExceptionMapper, ErrorResponse, InternalErrorResponse, ExceprionMessage.

В этом пакете созданы различные виды Exception, классы обработки ошибок, сообщения, которые выдаются в формате JSON, в зависимости от той или иной ошибки. Этот пакет помогает стандартизировать формат выдачи ошибок клиенту.

Описание всех классов проекта указаны в Приложении 4.

3.2.1.2. Взаимодействие и хранение данных в базе данных

В качестве базы данных используется СУБД PostgreSQL 12.

База данных состоит 12 таблиц, которые указаны в Приложении 2.

Для взаимодействия с базой данных используется платформенно независимый стандарт JDBC. Для реализации концепции ORM и хранения Java-обьектов в БД в удобном виде использована спецификация JPA. Для ее реализации использовалась библиотека Hibernate, которая описывает отношения между Java-объектами и записями в БД. Она предоставляет средства для автоматической генерации и обновления данных, автоматизирует построение SQL-запросов и обработки полученных наборов данных, что значительно уменьшает время разработки.

```
@Repository
public interface UserDao extends JpaRepository<UserEntity, Long> {
   @Query("SELECT u FROM RequestEntity AS ur " +
         "LEFT JOIN UserEntity AS u ON ur.userID = u.id " +
         "WHERE ur.courseID=?1 and ur.roleID=?2 and ur.approved=?3")
   List<UserEntity> getListOfUsers(Integer courseID, Integer roleID, Boolean approved);

   Optional<UserEntity> findByEmail(String email);
```

Листинг 3 - язык Java, представлено взаимодействие с таблицей БД Users

Для представления модели базы данных использовались сущности в JPA — представляющие данные, которые могут быть сохранены в БД. Сущность представляет собой таблицу, хранящуюся в базе данных. Каждое поле сущности представляет строку в таблице.

1. UserEntity — информация о пользователе.
2. ProfileEntity — данные о профиле пользователя.
3. RoleEntity — данные о ролях пользователя.
4. CourseEntity — данные о курсах.
5. PostEntity — данные о постах/обьявлениях.
6. TaskEntity — данные о заданиях.
7. ChatEntity — данные о чатах курса.
8. ChatMemberEntity — данные о участниках чата.
9. MessageEntity — данные о сообщении пользователя.
10. RequestEntity — данные о заявках пользователей.
11. UserCourseEntity — данные о участниках курса(пользователях).
12. UserTaskEntity— данные об ответе и оценке пользователя на задание.

```
@Entity
@Table(name = "users", schema = "public")
public class UserEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    @Column(name = "commonname")
    private String commonname;

    @NotNull
    @Column(name = "email", unique = true)
    private String email;

    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
    @Column(name = "created_at")
    private LocalDateTime createdAt;
```

Листинг 4 - язык Java, сущность User, описывающая таблицу Users

3.3. Описание и обоснование метода организации входных и выходных данных

3.3.1. Описание метода организации входных и выходных данных

Информационный обмен в Системе обеспечивается с помощью современных протоколов и форматов передачи данных. Для организации информационного обмена используются только документированные программные интерфейсы (API). Не допускается прямой вызов функций и данных одной подсистемы из другой, минуя документированный интерфейс или протокол обмена.

Информационный обмен между подсистемами Системы включает в себя сетевой режим передачи информации, предполагающий использование каналов связи Интернет, в которых данные передаются по протоколам HTTP/HTTPS.

Непосредственное взаимодействие между Backend-частью и клиентом происходит путем обмена данными через запрос-ответ.

Запрос с клиентской стороны включает в себя URL (endpoint API), определяющий затронутый ресурс, метод, определяющий требуемое действие (получить, изменить, создать, удалить ресурс) и может включать дополнительную информацию, закодированную в параметрах URL или данные в теле запроса (HTTP POST). Дополнительные данные могут быть включены в заголовки запроса (токены авторизации и аутентификации, типы контента и т. д.).

Методы поддерживают следующие типы запросов:

GET — тип запроса для получения ресурса.

POST — тип запроса для создания ресурса.

PUT — тип запроса для обновления ресурса.

DELETE — тип запроса для удаления ресурса.

Описание методов REST API приведено в Приложении 3.

Для передачи данных со стороны пользовательского интерфейса используется представление данных в формате JSON. Для передачи медиаконтента (изображения, видео-/аудиофайлы, документы в формате PDF или MS Office) в запросах используются файлы соответствующих типов.

Выходные данные формируются в ответ на запросы клиентской части Системы.

Для выходных данных используется:

1. информация, формируемая в пределах обработчиков запросов (уровень бизнес-логики).
2. информация из БД, которая дополнительно обрабатывается в соответствии с бизнес-логикой приложения.

Для унификации интерфейса клиент-серверного взаимодействия выходные данные должны быть преобразованы в JSON и соответствовать стандарту REST API.

Для каждого запроса Backend-часть отсылает в ответе статус обработки запроса: в случае успеха - 200 или 201, в случае неудачи - 400, 401, 404 и т. д. Также сообщение ошибки в формате JSON.

3.3.2. Обоснования выбора метода организации входных и выходных данных

В качестве входных и выходных данных был выбран формат JSON (JavaScript Object Notation).

Такой текстовый формат был выбран по следующим причинам:

1. Размер документа формата JSON значительно меньше, чем его аналоги.
2. Для большинства языков программирования его легко преобразовать в структуру данных.
3. Функционал для чтения и редактирования JSON-формата присутствует во многих языках программирования.

3.4. Описание и обоснование выбора состава технических и программных средств

3.4.1. Состав технических средств

Для надежной работы ПО Backend-подсистемы проекта требуется выделенное серверное оборудование, имеющее доступ к сети Интернет. Организация отдельного сервера может быть выполнена как на физическом уровне, так и на логическом (виртуальный сервер).

Минимальные технические характеристики сервера:

1. Вычислительная мощность – не менее одного процессора со спецификацией: тактовая частота 3,1 ГГц, 2 ядра, 4 потока;
2. Оперативная память – не менее 4Гб;
3. Жесткие диски – рабочее пространство не менее 100 Гб;
4. Сетевой адаптер – не менее 1 порта Gigabit Ethernet.

Технические средства должны быть масштабируемыми по производительности, емкости оперативной памяти, емкости дискового пространства и числу каналов ввода–вывода.

3.4.2. Состав программных средств

1. Наличие Java SE Development Kit 17 и выше.
2. Наличие фреймворков Spring Boot, Spring Data JPA, Spring Security, Postgresql Driver, Liquidbase, Slf4j.
3. ОС Ubuntu 21 и выше.
4. PostgreSQL версии 12 и выше.

3.4.3. Обоснование выбора технических и программных средств

Для реализации серверной части были выбраны язык программирования Java и фреймворк Spring Boot. Это позволило облегчить и ускорить процесс разработки, делая его более продуктивным и надежным. Также фреймворк предоставил гибкую настройку, надежную пакетную обработку, большое количество плагинов для взаимодействия с БД.

Дополнительно для реализации REST API использована библиотека Spring Data JPA, которая позволила гибко настроить взаимодействие с моделями базы данных.

Для решения задачи безопасности была выбрана библиотека Spring Security, которая предоставляет комплексные решения безопасности, аутентификации и авторизации.

Система PostgreSQL была выбрана в качестве СУБД в проекте. Такое решение было принято по нескольким причинам:

1. Открытый исходный код
2. Безопасность и надежность
3. Высокая масштабируемость
4. Возможность обработки сложных типов данных
5. Поддержка стандарта SQL

Для управления версиями базы данных была выбрана система Liquidbase. Это позволило делегировать автоматическое выполнение скриптов (добавление/изменение таблиц, столбцов БД) этой библиотеке при развертывании сервера.

В качестве тестового сервиса для деплоя серверноӣ части был выбран облачный VPS, на котором было установлено программное окружение в минимальных требованиях сервера. Это позволило не ограничиваться лимитами сервисов других платформ.

Для документации API и подключения Swagger использовалась библиотека Springdoc-api.

4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

4.1. Ориентировочная экономическая эффективность

В рамках данной работы расчет экономической эффективности не предусмотрен.

4.2. Предполагаемая потребность

Реализация платформы обмена информацией будет способствовать доступности, прозрачности, актуальности и достоверности информации в рамках учебного процесса НИСа.

Система должна стать инструментом информационной поддержки обучающихся в рамках НИСа, эффективного управления и использования информационных ресурсов, формирования удобной среды общения для всех участников учебного процесса.

4.3. Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами

В рамках изучения конкурентов этого приложения были выделены последующие главные свойства для сопоставления:

1. Функциональность – наличие главных функций приложения.
2. Кроссплатформенность – портируемость приложения на различные платформы.
3. Удобство использования преподавателем – удобство взаимодействия с приложением целевой аудиторией.
4. Безопасность данных – защищенность личных сведений.
5. Пользовательский интерфейс (UI) – интуитивность пользовательского интерфейса.
6. Доступность – политика цен приложения.
7. Размер используемой базы – количество потенциально зарегистрированных пользователей.

Процесс поиска аналогов в сети Интернет позволил выявить следующих прямых конкурентов:

- ВКонтакте — российская социальная сеть. [15]
- Campus Group - платформа частного сообщества, объединяющая студенческие организации, факультеты и все группы в кампусе [16]
- Society App – Приложение для управления событиями университета и организация задач. [17]

- LearningApps - онлайн-сервис, позволяющий создавать интерактивные упражнения для проверки знаний. [18]

- Online Test Pad – это образовательный онлайн-сервис для создания тестов, опросников, кроссвордов, логических игр и комплексных заданий. [19]

- ClassMarker – сайт для создания онлайн-тестирования. [20]

- Quizziz – платформа для создания викторин, тестов, общения с преподавателями. [21]

- Coursera - образовательная платформа, которая сотрудничает с ведущими университетами и организациями по всему миру, и предлагает онлайн курсы для всех. [22]

Приведена таблица сравнительного анализа конкурентов:

Критерии / Название приложения	ВКонтакте [16]	Campus Group [17]	Society App [18]	LearningApps [19]	Online Test Pad [20]	ClassMarker [21]	Quizizz [22]	Coursera [23]	HSE Apple
Функциональность	5	3	3	4	4	2	4	5	4
Кроссплатформенность	5	4	4	2	2	3	3	5	5
Удобство использования	4	4	4	4	4	4	3	5	5
Безопасность данных	4	4	4	3	3	3	4	5	4
UI	5	4	3	4	4	2	4	5	5
Доступность	5	2	2	3	3	3	4	5	4
Размер базы данных	4	3	2	3	3	1	2	4	3
Оценка:	**4,57**	**3,57**	**3,29**	**3,29**	**3,14**	**2,57**	**3,43**	**4,86**	**4,29**

Указанные выше характеристики ранжированы. Наивысший балл - 7, наименьший - 1.

5. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

2) ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

3) ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

4) ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

5) ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

7) ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

8) ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

9) ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

10) Системные требования Java [Электронный ресурс]// URL https://www.java.com/ru/download/help/sysreq.html (Дата обращения: 20.04.2022, режим доступа: свободный).

11) Системные требования PostgreSQL [Электронный ресурс]// URL https://postgrespro.ru/docs/postgresql/13/install-requirements (Дата обращения: 20.04.2022, режим доступа: свободный).

12) Требования к системе для Spring [Электронный ресурс]// URL: https://java-ru-blog.blogspot.com/2020/02/spring-boot-features.html (Дата обращения: 20.04.2022, режим доступа: свободный).

13) Документация SpringBoot [Электронный ресурс]// URL: https://spring.io/projects/spring-boot (Дата обращения: 20.04.2022, режим доступа: свободный).

14) Документация LiquidBase [Электронный ресурс]// URL: https://www.liquibase.org (Дата обращения: 20.04.2022, режим доступа: свободный).

15) ВКонтакте [Электронный ресурс]. Режим доступа: https://vk.com/ (Дата обращения 20.04.2022)

16) The all-in-one campus experience management platform // Campus Group [Электронный ресурс]. Режим доступа: https://www.campusgroups.com/ (Дата обращения 20.04.2022).

17) Skyrocket your society // Society App [Электронный ресурс]. Режим доступа: https://www.createyoursociety.com/ (Дата обращения 20.04.2022).

18) LearningApps [Электронный ресурс]. Режим доступа: https://learningapps.org/

(Дата обращения 20.04.2022)

19) Online test pads [Электронный ресурс]. Режим доступа: https://onlinetestpad.com/ (Дата обращения 20.04.2022)

20) ClassMarker [Электронный ресурс]. Режим доступа: https://www.classmarker.com/online-testing/faq/ (Дата обращения 20.04.2022)

21) Quizziz [Электронный ресурс]. Режим доступа: https://quizizz.com (Дата обращения 20.04.2022)

22) Coursera [Электронный ресурс]. Режим доступа: https://www.coursera.org/ (Дата обращения 20.04.2022)

ПРИЛОЖЕНИЕ 1
ТЕРМИНОЛОГИЯ

1. **RESTful API** — это API, соответствующий принципам архитектурного стиля REST (от англ. Representational State Transfer — «передача состояния представления»). REST API используют запросы HTTP для выполнения стандартных функций базы данных, таких как создание, чтение, обновление и удаление записей.

2. **Система управления базами данных (СУБД)** - совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

3. **HTTP** — протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных.

4. **JDBC –** платформенно независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета java.sql, входящего в состав Java SE.

5. **JPA –** спецификация API Java EE, предоставляет возможность сохранять в удобном виде Java-объекты в базе данных..

6. **JSON (**англ. *JavaScript Object Notation*) — текстовый формат обмена данными, основанный на JavaScript..

7. **JWT** (англ. *JSON Web Token*) — это JSON объект, который определен в открытом стандарте RFC 7519.

8. **Spring Boot**— среда с открытым исходным кодом, основанная на JAVA среда, которая используется для создания микросервиса

9. **База данных** (БД) — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе.API [11] - представляет собой набор правил, определяющих способ взаимодействия между приложениями или устройствами.

ПРИЛОЖЕНИЕ 2
СТРУКТУРА БАЗЫ ДАННЫХ

Таблица USER_ROLE

Таблица привилегий пользователей

Поле	**Описание**	**Тип значения**	**Значение по умолчанию**	**Свойство поля**
id	Уникальный идентификатор записи в таблице	smallint		pk, serial
role	Наименование привилегии в системе	varchar(50)		

Таблица COURSE

Таблица курсов

Поле	**Описание**	**Тип значения**	**Значение по умолчанию**	**Свойство поля**
id	Уникальный идентификатор записи в таблице	bigint		pk, serial
title	Наименование курса	varchar(50)		
description	Описание курса	text		nullable

Таблица USERS

Таблица пользователей

Поле	**Описание**	**Тип значения**	**Значение по умолчанию**	**Свойство поля**
id	Уникальный идентификатор записи в таблице	bigint		pk, serial
commonname	ФИО пользователя	varchar(50)		
email	email пользователя	varchar(50)		index
created_at	Дата и время создания записи	Timestamp without timezone(0)		

Таблица USER_COURSE

Таблица связей пользователей и курсов в системе

Поле	**Описание**	**Тип значения**	**Значение по умолчанию**	**Свойство поля**
id	Уникальный идентификатор записи в таблице	bigint		pk, serial
userid	ID пользователя внешний ключ из таблицы User, поле id	bigint		fk, index
courseid	ID курса внешний ключ из таблицы Course, поле id	bigint		fk, index
roleid	ID привилегии внешний ключ из таблицы User_role, поле id	smallint		fk, index

Таблица CHAT

Таблица групп (чатов) связанных с курсами

Поле	**Описание**	**Тип значения**	**Значение по умолчанию**	**Свойство поля**
id	Уникальный идентификатор записи в таблице	bigint		pk, serial
courseid	ID курса внешний ключ из таблицы Course, поле id	bigint		fk, index
title	Наименование группы (чата)	varchar(50)		
description	Описание группы (чата)	text		nullable
group_avatar	Ссылка на файл с изображением для аватара группы	text		nullable
created_at	Дата и время создания записи	Timestamp without timezone(0)		
updated_at	Дата и время изменения записи	t Timestamp without timezone(0)		nullable

Таблица CHAT_MEMBER

Таблица участников группы

Поле	**Описание**	**Тип значения**	**Значение по умолчанию**	**Свойство поля**
id	Уникальный идентификатор записи в таблице	bigint		pk, serial
chatid	ID группы внешний ключ из таблицы Chat, поле id	bigint		fk, index
userid	ID пользователя внешний ключ из таблицы User, поле id	bigint		fk, index
created_at	Дата и время создания записи	Timestamp without timezone(0)		
updated_at	Дата и время изменения записи	Timestamp without timezone(0)		nullable

Таблица CHAT_MESSAGE

Таблица сообщений в группе

Поле	**Описание**	**Тип значения**	**Значение по умолчанию**	**Свойство поля**
id	Уникальный идентификатор записи в таблице	bigint		pk, serial
chatid	ID группы внешний ключ из таблицы Chat, поле id	bigint		fk, index
userid	ID пользователя внешний ключ из таблицы User, поле id	bigint		fk, index
replyto	ID сообщения, на которое отвечает текущее сообщение внешний ключ из этой таблицы, поле id	bigint		fk
message	Текст сообщения	text		
media_link	Ссылка на прикрепленный к сообщению файл с изображением	text		nullable
doc_link	Ссылка на прикрепленный к сообщению файл с документом	text		nullable
created_at	Дата и время создания записи	Timestamp without timezone(0)		
updated_at	Дата и время изменения записи	Timestamp without timezone(0)		nullable

Таблица POST

Таблица постов ленты, связанных с курсом

Поле	**Описание**	**Тип значения**	**Значение по умолчанию**	**Свойство поля**
id	Уникальный идентификатор записи в таблице	bigint		pk, serial
courseid	ID курса внешний ключ из таблицы Course, поле id	bigint		fk, index
createdby	ID пользователя, создавшего запись внешний ключ из таблицы User, поле id	bigint		fk, index
updatedby	ID пользователя, изменившего запись внешний ключ из таблицы User, поле id	bigint		fk, index
title	Наименование сообщения для ленты	varchar (200)		
content	Текст сообщения для ленты	text		
media_link	Ссылка на прикрепленный к сообщению файл с изображением	text		nullable
doc_link	Ссылка на прикрепленный к сообщению файл с документом	text		nullable
created_at	Дата и время создания записи	Timestamp without timezone(0)		
updated_at	Дата и время изменения записи	Timestamp without timezone(0)		nullable

Таблица TASK

Таблица заданий для курса в системе

Поле	**Описание**	**Тип значения**	**Значение по умолчанию**	**Свойство поля**
id	Уникальный идентификатор записи в таблице	bigint		pk, serial
courseid	ID курса внешний ключ из таблицы Course, поле id	bigint		fk, index
form	Тип задания: “test” - лаб.работа, “lab” - контрольный тест	varchar (50)		
title	Наименование задания	varchar (150)		
description	короткое описание задания	text	null	nullable
task_content	Текст задания	text		
deadline	Срок выполнения задания	Timestamp without timezone(0)		nullable
status	Статус актуальности задания	boolean	'false'	
createdby	ID пользователя, создавшего запись внешний ключ из таблицы User, поле id	bigint		fk, index
updatedby	ID пользователя, изменившего запись внешний ключ из таблицы User, поле id	bigint		fk, index
created_at	Дата и время создания записи	Timestamp without timezone(0)		
updated_at	Дата и время изменения записи	Timestamp without timezone(0)		nullable

Таблица USER_TASK

Таблица связей заданий и пользователей

Поле	**Описание**	**Тип значения**	**Значение по умолчанию**	**Свойство поля**
id	Уникальный идентификатор записи в таблице	bigint		pk, serial
taskid	ID задания внешний ключ из таблицы Task, поле id	bigint		fk, index
userid	ID пользователя внешний ключ из таблицы User, поле id	bigint		fk, index
answer	Ответ пользователя	text		nullable
score	Оценка ответа	smallint	'0'	
status	Статус ответа	boolean	'false'	
createdby	ID пользователя, создавшего запись внешний ключ из таблицы User, поле id	bigint		fk, index
updatedby	ID пользователя, изменившего запись внешний ключ из таблицы User, поле id	bigint		fk, index
created_at	Дата и время создания записи	Timestamp without timezone(0)		
updated_at	Дата и время изменения записи	Timestamp without timezone(0)		nullable

Таблица USER_PROFILE - таблица профилей пользователей в системе

Поле	**Описание**	**Тип значения**	**Значение по умолчанию**	**Свойство поля**
id	Уникальный идентификатор записи в таблице	bigint		pk, serial
userid	ID пользователя внешний ключ из таблицы User, поле id	bigint		fk, index
avatar	Ссылка на файл с изображением для аватара	text		nullable
created_at	Дата и время создания записи	Timestamp without timezone(0)		
updated_at	Дата и время изменения записи	Timestamp without timezone(0)		nullable

Таблица USER_REQUEST_COURSE

Таблица заявок пользователей на курсы

Поле	**Описание**	**Тип значения**	**Значение по умолчанию**	**Свойство поля**
id	Уникальный идентификатор записи в таблице	bigint		pk, serial
userid	ID пользователя внешний ключ из таблицы User, поле id	bigint		fk, index
courseid	ID курса внешний ключ из таблицы Course, поле id	bigint		fk, index
roleid	ID привилегии внешний ключ из таблицы User_role, поле id	smallint		fk, index
approved	Статус заявки (одобрено, отклонено)	boolean	'false' (отклонено)	
created_at	Дата и время создания записи	Timestamp without timezone(0)		
updated_at	Дата и время изменения записи	Timestamp without timezone(0)		nullable

ПРИЛОЖЕНИЕ 3
ОПИСАНИЕ REST API

Header: Authorization: "access_token"

Авторизация

GET /auth

Запрос авторизации пользователя

Ответ на успешный запрос:

В теле ответа передается объект User в JSON формате

Content-Type response header: application/json

Authorization header: "access_token"

Пример ответа:

Content-Type: application/json

```
{
  "commonname" : "firstName",
  "createdAt" : "2000-01-23 14:32:54",
  "id" : 0,
  "email" : "dayana@gmail.com"
}
```

Коды ответа на запрос:

Код ответа	**Сообщение**	**Описание**
200	OK	Запрос обработан успешно
401	Invalid Code	Неверные данные в запросе
403	Access Denied	Для данного аккаунта доступ запрещен

Курсы

Пользователи

GET /user

Получить список объектов с данными пользователей курса

Параметры Query:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
courseID	integer		ID курса
roleID	integer		ID привилегии пользователя
approved	boolean		статус заявки пользователя

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается массив объектов с параметрами заявки и пользователя: статус заявки, дата создания заявки, идентификатор заявки, идентификатор курса, объект User.

Пример ответа:

Content-Type: application/json

```
[ {
  "commonname" : "firstName",
  "createdAt" : "2000-01-23 14:32:54",
  "id" : 0,
  "email" : "dayana@gmail.com"
  },
{
  "commonname" : "firstName",
  "createdAt" : "2000-01-23 14:32:54",
  "id" : 1,
  "email" : "dayana@gmail.com"
} ]
```

GET /user/{userID}/application/approved

Получить пользователя курса по идентификатору пользователя {userID}

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
userID	long	обязательный	ID пользователя

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается объект User

Пример ответа:

Content-Type: application/json

```
{
"commonname" : "firstName",
  "createdAt" : "2000-01-23 14:32:54",
  "id" : 1,
  "email" : "dayana@gmail.com"
}
```

PUT /request/{courseID}/{userID}

Изменить заявку пользователя курса

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
courseID	integer	обязательный	ID курса
userID	long	обязательный	ID пользователя

Данные для изменения заявки передаются в теле запроса в формате JSON

Content-Type request header: application/json

Request body

Параметры тела запроса формата JSON

Параметр запроса	Тип параметра	Обязательный параметр	Описание
approved	boolean	обязательный	Статус заявки

Пример: { "approved" : true }

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается объект User_request_course

Пример ответа:

Content-Type: application/json

```
{
  "createdAt" : "2000-01-23 04:56:07",
  "approved" : false,
  "id" : 0,
  "roleID" : 2,
  "userID" : 1,
  "courseID" : 5,
  "updatedAt" : "2000-01-23 04:56:07"
}
```

Группы

GET /courses/{courseID}/group

Получить список существующих объектов Group для курса. Каждый объект содержит свойства группы:

1. id
2. courseID
3. title
4. description
5. group_avatar
6. status
7. createdBy

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
courseID	integer	обязательный	ID курса

Параметры Query:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
status	boolean		**статус группы**

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается массив объектов с параметрами объекта Group:

статус группы, дата создания группы, идентификатор группы, идентификатор создателя группы, наименование группы, идентификатор курса, описание группы, ссылка на файл аватара группы.

Пример ответа:

Content-Type: application/json

```
[ {
  "createdAt" : "createdAt",
  "createdBy" : 1,
  "description" : "description",
  "id" : 0,
  "status" : true,
  "title" : "title",
  "courseID" : 6,
  "group_avatar" : "group_avatar"
}, {
  "createdAt" : "createdAt",
  "createdBy" : 1,
  "description" : "description",
  "id" : 0,
  "status" : true,
  "title" : "title",
  "courseID" : 6,
```

```
  "group_avatar" : "group_avatar"
} ]
```

GET /group/{groupID}

Получить объект существующей группы для курса

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
groupID	integer	обязательный	ID группы

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается объект Group.

Пример ответа:

Content-Type: application/json

```
{
  "createdAt" : "2000-01-23 04:56:07",
  "updatedBy" : 5,
  "createdBy" : 1,
  "description" : "description",
  "id" : 0,
  "state" : true,
  "title" : "title",
  "courseID" : 6,
  "group_avatar" : "http://example.com/aeiou",
  "updatedAt" : "2000-01-23 04:56:07"
}
```

DELETE /group/{groupID}

Удалить группу курса по идентификатору ID группы {groupID}

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
groupID	integer	обязательный	ID группы

Коды ответа на запрос:

Код ответа	Сообщение	Описание
200	OK	Запрос обработан успешно
401	Invalid Code	Неверные данные в запросе
403	Access Denied	Для данного аккаунта доступ запрещен
404	Course Not Found	Курс не найден в базе данных

Задания

GET /course/{courseID}/task

Получить список существующих объектов Task для курса.

Каждый объект содержит свойства задания:

1. id
2. courseID
3. title
4. deadline
5. status
6. createdBy
7. createdAt

Критерии фильтра: ID задания, начиная с которого прислать список

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание

Параметры Query:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
start	long		ID задания

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается массив объектов Task с параметрами:

id, courseID, title, deadline, state, createdBy, createdAt

Пример ответа:

Content-Type: application/json


```
[ {
  "createdAt" : "2000-01-23T04:56:07.000+00:00",
  "createdBy" : 1,
  "id" : 0,
  "status" : true,
  "type" : true,
  "title" : "title",
  "deadline" : "2000-01-23T04:56:07.000+00:00",
  "courseID" : 6
}, {
  "createdAt" : "2000-01-23T04:56:07.000+00:00",
  "createdBy" : 1,
  "id" : 0,
  "status" : true,
  "type" : true,
  "title" : "title",
  "deadline" : "2000-01-23T04:56:07.000+00:00",
  "courseID" : 6
} ]
```


GET /course/{courseID}/task/{taskID}

Получить существующий объект Task для курса по идентификатору taskID

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
courseID	integer	обязательный	ID курса
taskID	integer	обязательный	ID задания

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается объект Task

Пример ответа:

Content-Type: application/json


```
{
  "createdAt" : "2000-01-23T04:56:07.000+00:00",
  "updatedBy" : 5,
  "createdBy" : 1,
  "description" : "description",
  "task_content" : "task_content",
  "id" : 0,
  "state" : false,
  "type" : true,
  "title" : "title",
  "deadline" : "2000-01-23T04:56:07.000+00:00",
  "courseID" : 6,
  "updatedAt" : "2000-01-23T04:56:07.000+00:00"
}
```


GET /course/{courseID}/answer

Получить список существующих объектов User_task для курса.

Каждый объект содержит свойства ответа пользователя:

1. id
2. courseID
3. taskID
4. userID
5. answer
6. score
7. state

8. createdAt

Критерии фильтра: статус ответа пользователя, ID задания, тип задания (лаб.работа, тест), ID пользователя

Параметры Path:

Параметр запроса	**Тип параметра**	**Обязательный параметр**	**Описание**
courseID	integer	обязательный	ID курса

Параметры Query:

Параметр запроса	**Тип параметра**	**Обязательный параметр**	**Описание**
state_answer	boolean		статус задания пользователя
userID	integer		ID пользователя
type	boolean		тип задания
taskID	integer		ID задания

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается массив объектов Task и User_task

Пример ответа:

Content-Type: application/json


```
[
 {
   "score" : 1,
   "createdAt" : "2000-01-23T04:56:07.000+00:00",
   "updatedBy" : 5,
   "answer" : "answer",
   "createdBy" : 5,
   "id" : 0,
   "state" : false,
```

```
    "taskID" : 6,
    "updatedAt" : "2000-01-23T04:56:07.000+00:00"
},
  {
    "score" : 1,
    "createdAt" : "2000-01-23T04:56:07.000+00:00",
    "updatedBy" : 5,
    "answer" : "answer",
    "createdBy" : 5,
    "id" : 0,
    "state" : false,
    "taskID" : 6,
    "updatedAt" : "2000-01-23T04:56:07.000+00:00"
  }
```


DELETE /course/{courseID}/task/{taskID}

Удаление существующего задания курса {courseID} по идентификатору ID {taskID}

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
courseID	integer	обязательный	ID курса
taskID	integer	обязательный	ID задания

Коды ответа на запрос:

Код ответа	Сообщение	Описание
200	OK	Запрос обработан успешно
401	Invalid Code	Неверные данные в запросе
403	Access Denied	Для данного аккаунта доступ запрещен
404	Task Not Found	Задание не найдено в базе данных

Пользователи

GET /user

Получить список существующих пользователей.

Список пользователей можно отфильтровать по критериям:

1. курс пользователя
2. привилегия пользователя
3. статус заявки

Параметры Query:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
courseID	integer		ID курса
approved	boolean		Статус заявки
roleID	integer		ID привилегии пользователя

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается массив объектов User

Пример ответа:
Content-Type: application/json

```
[ {
  "commonname" : "firstName",
  "createdAt" : "2000-01-23",
  "roleId" : 6,
  "id" : 0,
  "email" : ""
}, {
  "commonname" : "firstName",
  "createdAt" : "2000-01-23",
  "roleId" : 6,
  "id" : 0,
  "email" : ""
} ]
```

GET /users/{userID}

Получение данных пользователя по ID.

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
userID	integer	обязательный	ID пользователя

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается объект User

Пример ответа:

Content-Type: application/json

```
{
  "commonname" : "firstName",
  "id" : 0,
  "email" : "email",
  "createDate" : "2000-01-23"
}
```

PUT /users/{userID}

Обновление данных существующего объекта User

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
userID	integer	обязательный	ID пользователя

Данные для изменения параметров пользователя передаются в теле запроса в формате JSON

Content-Type request header: application/json

Request body

Параметры тела запроса формата JSON

Параметр запроса	Тип параметра	Обязательный параметр	Описание
commonname	string		ФИО пользователя
email	string		Email пользователя
roleID	integer		ID привилегии пользователя

Пример: { "value" : { "firstName" : "НовоеИмя" } }

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается обновленный объект User

Пример ответа:

Content-Type: application/json

```
{
  "commonname" : "firstName",
  "createdAt" : "2000-01-23",
  "roleId" : 6,
  "id" : 0,
  "email" : ""
}
```

Коды ответа на запрос:

Код ответа	Сообщение	Описание
200	OK	Запрос обработан успешно
400	Missing Required Information	Неверные данные в запросе
404	User Not Found	Акаунта не найден

Профиль пользователя

GET /profile/{userID}

Получение данных профиля существующего пользователя

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
userID	integer	обязательный	ID пользователя

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается объект User_Profile

Пример ответа:

Content-Type: application/json


```
{
  "notification" : true,
  "createdAt" : "2000-01-23T04:56:07.000+00:00",
  "language" : true,
  "id" : 0,
  "avatar" : "http://example.com/aeiou",
  "userID" : 6,
  "updatedAt" : "2000-01-23T04:56:07.000+00:00"
}
```


Коды ответа на запрос:

Код ответа	Сообщение	Описание
200	OK	Запрос обработан успешно
401	Unauthorized	Для запроса требуется авторизация
404	User Not Found	UserID не найден в базе данных

PUT /profile/{userID}

Обновление данных профиля существующего пользователя

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
userID	integer	обязательный	ID пользователя

Данные для изменения профиля пользователя передаются в теле запроса в формате JSON

Content-Type request header: application/json

Request body

Параметры тела запроса формата JSON

Параметр запроса	Тип параметра	Обязательный параметр	Описание
notification	boolean		Статус уведомлений
group_avatar	string		Файл с изображением
language	boolean		Язык интерфейса

Пример: { "value" : { "notification" : false, "language" : false } }

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается новый объект User_profile

Пример ответа:

Content-Type: application/json

```
{
  "notification" : true,
  "createdAt" : "2000-01-23T04:56:07.000+00:00",
  "language" : true,
  "id" : 0,
  "avatar" : "http://example.com/aeiou",
  "userID" : 6,
  "updatedAt" : "2000-01-23T04:56:07.000+00:00"
}
```

Коды ответа на запрос:

Код ответа	Сообщение	Описание
200	OK	Запрос обработан успешно
400	Missing Required Information	Неверные данные в запросе
404	User Not Found	Акаунт не найден

Задания

PUT /courses/{courseID}/tasks/{taskID}

Изменить свойства объекта Task. Для изменения свойства задания передаются в теле запроса в формате JSON :

1. курс задания: courseID
2. тип задания: type,

3. наименование задания: title,
4. краткое описание задание: description,
5. содержимое задания: task_content,
6. срок выполнения задания: deadline,
7. статус задания: state

Параметры Path:

Параметр запроса	**Тип параметра**	**Обязательный параметр**	**Описание**
courseID	integer	обязательный	ID курса
taskID	integer	обязательный	ID задания

Данные для изменения задания передаются в теле запроса в формате JSON

Обязательные требования к параметрам в теле запроса:

минимум один из перечисленных параметров должен присуствовать.

Content-Type request header: application/json

Request body

Параметры тела запроса формата JSON

Параметр запроса	**Тип параметра**	**Обязательный параметр**	**Описание**
courseID	integer	обязательный	ID курса
type	boolean	обязательный	Тип задания
title	string	обязательный	Наименование задания
description	string		Описание задания
task_content	string	обязательный	Текст задания
deadline	string format: datetime	обязательный	Срок выполнения задания
state	boolean	обязательный	Статус задания

Пример: { "value" : { "courseID" : 1, "type" : false, "title" : "Контрольный тест 3", "description" : "Тест для 3 курса", "task_content" : "Задания для теста", "deadline" : "2022-02-28T14:15:22Z", "state" : true } }

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается обновленный объект Task

Пример ответа:

Content-Type: application/json


```
{
  "createdAt" : "2000-01-23T04:56:07.000+00:00",
  "updatedBy" : 5,
  "createdBy" : 1,
  "description" : "description",
  "task_content" : "task_content",
  "id" : 0,
  "state" : false,
  "type" : true,
  "title" : "title",
  "deadline" : "2000-01-23T04:56:07.000+00:00",
  "courseID" : 6,
  "updatedAt" : "2000-01-23T04:56:07.000+00:00"
}
```


Пример ответа:

POST /courses/{courseID}/tasks

Создать задание для курса

Параметры Path:

Параметр запроса	**Тип параметра**	**Обязательный параметр**	**Описание**
courseID	integer	обязательный	ID курса

Данные нового задания передаются в теле запроса в формате JSON

Content-Type request header: application/json

Request body

Параметры тела запроса формата JSON

Параметр запроса	Тип параметра	Обязательный параметр	Описание
courseID	integer	обязательный	ID курса
type	boolean	обязательный	Тип задания
title	string	обязательный	Наименование задания
description	string		Описание задания
task_content	string	обязательный	Текст задания
deadline	string format: datetime	обязательный	Срок выполнения задания
state	boolean	обязательный	Статус задания

Пример: { "value" : { "courseID" : 0, "type" : true, "title" : "Лабораторная работа 2", "description" : "Лаб.работа для курса 2", "task_content" : "Задание", "deadline" : "2022-02-24T14:15:22Z", "state" : true } }

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается объект Task

Пример ответа:
Content-Type: application/json


```
{
  "createdAt" : "2000-01-23T04:56:07.000+00:00",
  "updatedBy" : 5,
  "createdBy" : 1,
  "description" : "description",
  "task_content" : "task_content",
  "id" : 0,
  "state" : false,
  "type" : true,
  "title" : "title",
  "deadline" : "2000-01-23T04:56:07.000+00:00",
  "courseID" : 6,
  "updatedAt" : "2000-01-23T04:56:07.000+00:00"
}
```


POST /courses/{courseID}/user_tasks

Создать ответ пользователя для задания

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
courseID	integer	обязательный	ID курса

Данные ответа для задания передаются в теле запроса в формате JSON

Content-Type request header: application/json

Request body

Параметры тела запроса формата JSON

Параметр запроса	Тип параметра	Обязательный параметр	Описание
courseID	integer	обязательный	ID курса
userID	integer	обязательный	ID пользователя
taskID	integer	обязательный	ID задания
answer	string	обязательный	Описание задания
state	boolean	обязательный	Статус ответа

Пример: { "value" : { "courseID" : 0, "userID" : 0, "taskID" : 0, "answer" : "Ответ на лаб.работу 1", "state" : true } }

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается объект User_task

Пример ответа:

Content-Type: application/json


```
{
  "score" : 1,
  "createdAt" : "2000-01-23T04:56:07.000+00:00",
  "updatedBy" : 5,
  "answer" : "answer",
  "createdBy" : 5,
  "id" : 0,
  "state" : false,
  "taskID" : 6,
```

```
  "updatedAt" : "2000-01-23T04:56:07.000+00:00"
}
```

Группы

Участники

GET /groups/{groupID}/members

Получить список объектов с данными участников группы с идентификатором группы {groupID}.

Критерии фильтрации: ID привилегии пользователя.

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
groupID	integer	обязательный	ID группы

Параметры Query:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
roleID	integer		ID привилегии пользователя

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается массив объектов с параметрами группы и пользователя: идентификатор группы, идентификатор записи в таблице Group_member, объект User.

Пример ответа:

Content-Type: application/json

```
[ {
  "groupID" : 6,
  "id" : 0,
  "user" : {
    "firstName" : "firstName",
    "lastName" : "lastName",
    "createdAt" : "2000-01-23",
    "roleId" : 6,
    "id" : 0,
    "email" : "email"
  }
```

```
}, {
  "groupID" : 6,
  "id" : 0,
  "user" : {
    "firstName" : "firstName",
    "lastName" : "lastName",
    "createdAt" : "2000-01-23",
    "roleId" : 6,
    "id" : 0,
    "email" : "email"
  }
```

GET /groups/{groupID}/members/{userID}

Получить данные участника группы

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
groupID	integer	обязательный	ID группы
userID	integer	обязательный	ID пользователя

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается объект с параметрами группы и пользователя: идентификатор группы, идентификатор записи в таблице Group_member, объект User.

Пример ответа:

Content-Type: application/json

```
{
  "groupID" : 6,
  "id" : 0,
  "user" : {
    "firstName" : "firstName",
    "lastName" : "lastName",
    "createdAt" : "2000-01-23T04:56:07.000+00:00",
    "roleId" : 5,
    "id" : 1,
    "email" : ""
  }
```

```
}
```

DELETE /groups/{groupID}/members/{userID}

Удалить пользователя из группы

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
groupID	integer	обязательный	ID группы
userID	integer	обязательный	ID пользователя

Коды ответа на запрос:

Код ответа	Сообщение	Описание
200	OK	Запрос обработан успешно
401	Invalid Code	Неверные данные в запросе
403	Access Denied	Для данного аккаунта доступ запрещен
404	User Not Found	Пользователь не найден в базе данных

Сообщения

GET /groups/{groupID}/messages

Получить список сообщений существующей группы

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
groupID	integer	обязательный	ID группы

Параметры Query:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
start	integer	обязательный	ID сообщения, с которого начинается выборка default: 0
limit	integer	обязательный	количество сообщений default: 20

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается массив сообщений в группе

Пример ответа:

Content-Type: application/json


```
[ {
  "createdAt" : "2000-01-23T04:56:07.000+00:00",
  "groupID" : 6,
  "doc_link" : "http://example.com/aeiou",
  "replyTo" : 5,
  "media_link" : "http://example.com/aeiou",
  "id" : 0,
  "message" : "message",
  "userID" : 1,
  "updatedAt" : "2000-01-23T04:56:07.000+00:00"
}, {
  "createdAt" : "2000-01-23T04:56:07.000+00:00",
  "groupID" : 6,
  "doc_link" : "http://example.com/aeiou",
  "replyTo" : 5,
  "media_link" : "http://example.com/aeiou",
  "id" : 0,
  "message" : "message",
```

```
  "userID" : 1,
  "updatedAt" : "2000-01-23T04:56:07.000+00:00"
} ]
```

GET /groups/{groupID}/messages/{messageID}

Получить объект Message существующей группы

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
groupID	integer	обязательный	ID группы
messageID	integer	обязательный	ID сообщения

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается объект Message

Пример ответа:

Content-Type: application/json


```
{
  "createdAt" : "2000-01-23T04:56:07.000+00:00",
  "groupID" : 6,
  "doc_link" : "http://example.com/aeiou",
  "replyTo" : 5,
  "media_link" : "http://example.com/aeiou",
  "id" : 0,
  "message" : "message",
  "userID" : 1,
  "updatedAt" : "2000-01-23T04:56:07.000+00:00"
}
```


POST /groups/{groupID}/messages

Создать сообщение для существующей группы

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
groupID	integer	обязательный	ID группы

Данные нового сообщения группы передаются в теле запроса в формате JSON

Content-Type request header:

- application/json
- multipart/form-data

Request body

Параметры тела запроса формата JSON

Параметр запроса	Тип параметра	Обязательный параметр	Описание
groupID	integer	обязательный	ID группы
userID	integer	обязательный	ID пользователя
replyTo	integer		ID сообщения, на которое текущее сообщение является ответом
message	string	обязательный	Текст сообщения

Пример: { "value" : { "groupID" : 0, "userID" : 0, "message" : "Привет всем!" } }

Параметры тела запроса формата FormObject

Параметр запроса	Тип параметра	Обязательный параметр	Описание
image	boolean	обязательный	Тип файла: true – изображение, false - документ
file	string format: binary	обязательный	Файл

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается объект Group_message

Пример ответа:

Content-Type: application/json


```
{
  "createdAt" : "2000-01-23T04:56:07.000+00:00",
  "groupID" : 6,
  "doc_link" : "http://example.com/aeiou",
  "replyTo" : 5,
  "media_link" : "http://example.com/aeiou",
  "id" : 0,
  "message" : "message",
  "userID" : 1,
  "updatedAt" : "2000-01-23T04:56:07.000+00:00"
}
```


DELETE /groups/{groupID}/messages/{messageID}

Удаление сообщения пользователя в группе

Параметры Path:

Параметр запроса	**Тип параметра**	**Обязательный параметр**	**Описание**
groupID	integer	обязательный	ID группы
messageID	integer	обязательный	ID сообщения

Коды ответа на запрос:

Код ответа	**Сообщение**	**Описание**
200	OK	Запрос обработан успешно
401	Invalid Code	Неверные данные в запросе
403	Access Denied	Для данного аккаунта доступ запрещен
404	Message Not Found	Сообщение не найдено в базе данных

Посты (лента постов)

GET /courses/{courseID}/posts

Получение списка объектов данных для постов ленты существующего курса, каждый объект

содержит: "id", "courseID", "title", "media_link", "createdAt".

Query параметры: limit, start - опциональны.

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
courseID	integer	обязательный	ID курса

Параметры Query:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
start	integer	обязательный	ID поста, с которого начинается выборка default: 0
limit	integer	обязательный	количество сообщений default: 20
unread	boolean		Статус прочтения

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается массив данных объекта Post

Пример ответа:
Content-Type: application/json

```
[ {
  "createdAt" : "createdAt",
  "media_link" : "media_link",
  "id" : 0,
  "title" : "title",
  "courseID" : 6,
  "updatedAt" : "updatedAt"
}, {
  "createdAt" : "createdAt",
  "media_link" : "media_link",
  "id" : 0,
  "title" : "title",
  "courseID" : 6,
  "updatedAt" : "updatedAt"
```

```
}, {
  "createdAt" : "createdAt",
  "media_link" : "media_link",
  "id" : 0,
  "title" : "title",
  "courseID" : 6,
  "updatedAt" : "updatedAt"
}
]
```


POST /courses/{courseID}/posts

Создание поста для существующего курса.

Параметры Path:

Параметр запроса	Тип параметра	Обязательный параметр	Описание
courseID	integer	обязательный	ID курса

Данные для нового поста курса передаются в теле запроса в формате JSON и FormObject.

Файл изображения или документа для поста передается в теле запроса с использованием FormData с указанием заголовка: Content-Type: multipart/form-data

Content-Type request header:

1. application/json
2. multipart/form-data

Request body

Параметры тела запроса формата JSON

Параметр запроса	Тип параметра	Обязательный параметр	Описание
courseID	integer	обязательный	ID курса
title	string	обязательный	Наименование поста
content	string	обязательный	Текст поста

Пример: { "value" : { "courseId" : 0, "title" : "Новый пост", "content" : "Текст поста для ленты" } }

Параметры тела запроса формата FormObject

Параметр запроса	Тип параметра	Обязательный параметр	Описание
image	boolean	обязательный	Тип файла: true – изображение, false - документ
file	string format: binary	обязательный	Файл

Ответ на успешный запрос:

Content-Type response header: application/json

В теле ответа в формате JSON передается новый объект Post

Пример ответа:

Content-Type: application/json

```
{ "createdAt" : "createdAt",
  "updatedBy" : 5,
  "createdBy" : 1,
  "doc_link" : "doc_link",
  "media_link" : "media_link",
  "id" : 0,
  "title" : "title",
  "courseId" : 6,
  "content" : "content",
  "updatedAt" : "updatedAt"
}
```

ПРИЛОЖЕНИЕ 4
ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ

Таблица 4.1

Описание и функциональное назначение классов серверной части

Класс	Назначение
UserController	Контроллер для управления аккаунтом пользователя.
UserService	Сервис для работы с аккаунтом пользователя.
UserDao	Интерфейс позволяет взаимодействовать с моделями пользователей.
UserEntity	Модель пользователя в БД
CourseController	Контроллер для управления информацией о курсе.
CourseService	Сервис для работы с курсом.
CourseDao	Интерфейс позволяет взаимодействовать с моделями курсов.

Класс	Назначение
CourseEntity	Модель курса в БД
TaskController	Контроллер для управления информацией о заданиях.
TaskService	Сервис для работы с заданиями.
TaskDao	Интерфейс позволяет взаимодействовать с моделями заданий.
TaskEntity	Модель задания в БД
PostController	Контроллер для управления информацией о постах.
PostService	Сервис для работы с постами.
PostDao	Интерфейс позволяет взаимодействовать с моделями постов.
PostEntity	Модель поста в БД
ChatController	Контроллер для управления информацией о чатах.
ChatService	Сервис для работы с чатами.
ChatDao	Интерфейс позволяет взаимодействовать с моделями чатов.
ChatEntity	Модель чата в БД
ChatMemberDao	Интерфейс позволяет взаимодействовать с моделями участников чатов.
ChatMemberEntity	Модель участника чата в БД.
MessageDao	Интерфейс позволяет взаимодействовать с моделями сообщений.
MessageEntity	Модель сообщения чата в БД.
ProfileDao	Интерфейс позволяет взаимодействовать с моделями профиля пользователей.
ProfileEntity	Модель профиля пользователей в БД.
RequestDao	Интерфейс позволяет взаимодействовать с моделями заявок пользователей.
RequestEntity	Модель заявки пользователя в БД.
RoleEntity	Модель роли пользователя в БД.
UserCourseDao	Интерфейс позволяет взаимодействовать с моделями пользователей на курсе.
UserCourseEntity	Модель пользователя на курсе в БД.
UserTaskDao	Интерфейс позволяет взаимодействовать с моделями ответов пользователей на задания.
UserTaskEntity	Модель ответа на задания пользователя в БД.

Продолжение таблицы 4.1

AppException	Главный класс exception приложения.
BusinessException	Класс ошибок бизнес-логики.
TechnicalException	Класс технических ошибок.
ErrorResponse	Класс, хранящий ответы-сообщения на ошибки
ExceptionMapper	Класс, обрабатывающий ошибки.
ExceptionMessage	Класс, хранящий константы-сообщения об ошибках.
OpenApiConfig	Класс конфигурации для документации API.
JWTAuthFilter	Класс для декодирования JWT токена и авторизации пользователя.
SecurityConfig	Класс конфигурации безопасности программы.
UserAndRole	Класс, хранящий информацию о пользователе и его ролях после декодирования токена.

HseappleApplication	Главный запускаемый класс.

ПРИЛОЖЕНИЕ 5
ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ПОЛЕЙ МЕТОДОВ И СВОЙСТВ

Таблица 5.1

Описание методов и свойств класса UserController.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
getUser	public	Метод	userID: Long	Метод для получения информации и пользователе.

updateUser	public	Метод	newUser: UserEntity, userID: Long	Метод для обновления информации о пользователе.
getListUsers	public	Метод	courseID, roleID: Integer, approved: Boolean	Метод для получения списка пользователей.
getRequest	public	Метод	courseID: Integer, userID: Long	Метод для получения заявки пользователя.
updateRequest	public	Метод	courseID: Integer, userID: Long, requestEntity: RequestEntity	Метод для обновления заявки пользователя.
getProfile	public	Метод		Метод для получения информации о профиле пользователя.
updateProfile	public	Метод	profileEntity: ProfileEntity	Метод для обновления информации о профиле пользователя.
createRequest	public	Метод	courseID, roleID: Integer	Метод для создания заявки пользователя.
auth	public	Метод		Метод для авторищации пользователя
Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
logger	private	Logger	get, set	Логгер.
userService	private	UserService	get, set	Сервис для работы с пользователями.

Таблица 5.2

Описание методов и свойств класса UserService

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
findUser	public	Метод	userID: Long	Метод для поиска пользователя.
changeUser	public	Метод	newUser: UserEntity, userID: Long	Метод для изменения

				информации и пользователе.
createUser	public	Метод	commonname, email: String	Метод для создания пользователя.
getUsers	public	Метод	courseID, roleID: Integer, approved: Boolean	Метод для получения списка пользователей.
getUserRequest	public	Метод	courseID: Integer, userID: Long	Метод для поиска заявки пользователя.
updateUserRequest	public	Метод	courseID: Integer, userID: Long, requestEntity: RequestEntity	Метод для изменения заявки пользователя.
getProfile	public	Метод		Метод для поиска информации о профиле пользователя.
createRequest	public	Метод	courseID, roleID: Integer	Метод для создания заявки пользователя.
createProfile	public	Метод	id: Long	Метод для создания профиля пользователя. встреча.
createUserCourse	public	Метод	courseID, roleID: Integer userID: Long	Метод для создания добавления пользователя в таблицу UserCourse.
registerUser	public	Метод		Метод для отправки авторизованного пользователя.
updateProfile	public	Метод	profileEntity: ProfileEntity	Метод для обновления профиля пользователя.

Свойства

Имя	Модификатор доступа	Тип	Доступ	Назначение
logger	private	Logger	get, set	Логгер.
userDao	private	UserDao	get, set	Репозиторий пользователей.
courseDao	private	CourseDao	get, set	Репозиторий курсов.
requestDao	private	RequestDao	get, set	Репозиторий заявок.
userCourseDao	private	UserCourseDao	get, set	Репозиторий пользователей курса.
profileDao	private	ProfileDao	get, set	Репозиторий профиля.

chatService	private	ChatService	get, set	Сервис для работы с чатами.

Таблица 5.3

Описание методов и свойств интерфейса UserDao.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
getListOfUsers	public	Метод	courseID, roleID: Integer, approved: Boolean	Метод для получения пользователей из БД.
findByEmail	public	Метод	email: String	Метод для получения пользователя по его email из БД.
findAllRoleById	public	Метод	id: Long	Метод для нахождения всех ролей пользователя по его ID из БД.
getProfile	public	Метод	id: Long	Метод для получения профиля пользователя по его ID из БД.

Таблица 5.4

Описание методов и свойств класса UserEntity.java

Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
id	private	Long	get, set	ID пользователя
commonname	private	String	get, set	ФИО пользователя
email	private	String	get, set	email пользователя
createdAt	private	LocalDateTime	get, set	Дата создания записи

Таблица 5.5

Описание методов и свойств класса CourseController.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
getCourses	public	Метод		Метод для получения информации о курсах
getListRequests	public	Метод	courseID: Integer, approved: Boolean	Метод для получения списка заявок на курс

Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
logger	private	Logger	get, set	Логгер.
courseService	private	CourseService	get, set	Сервис для работы с курсами

Таблица 5.6

Описание методов и свойств класса CourseService.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
findAllCourse	public	Метод		Метод для нахождения информации о курсах
findAllRequests	public	Метод	courseID: Integer, approved: Boolean	Метод для нахождения списка заявок на курс
Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
courseDao	private	CourseDao	get, set	Репозиторий курсов
requestDao	private	RequestDao	get, set	Репозиторий заявок пользователей

В интерфейсе CourseDao отсутствуют вручную написанные свойства и методы.

Таблица 5.7

Описание методов и свойств класса CourseEntity.java

Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
id	private	Integer	get, set	ID курса
title	private	String	get, set	Название курса
description	private	String	get, set	Описание курса

Таблица 5.8

Описание методов и свойств класса TaskController.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
getTaskForCourse	public	Метод	taskID: Long	Метод для получения задания на курсе
updateTask	public	Метод	newTask: TAskEntity, taskID: Long	Метод для обновления задание
deleteTask	public	Метод	taskID: Long	Метод для удаления задания

getTasks	public	Метод	courseID: Integer, start: Long	Метод для получения списка заданий
createTask	private	Метод	taskEntity: TaskEntity	Метод для создания задания
getAnswerTasks	private	Метод	taskID: Long, state_answer: Boolean, form: String	Метод для получения списка ответов на задание
createAnswer	private	Метод	taskID: Long, userTaskEntity: UserTaskEntity	Метод для создания ответа на задание
updateUserTask	private	Метод	newUserTask: UserTaskEntity	Метод для выставления оценки на задание
Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
logger	private	Logger	get, set	Логгер.
taskService	private	TaskService	get, set	Сервис для работы с заданиями

Таблица 5.9

Описание методов и свойств класса TaskService.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
getTaskForCourse	public	Метод	taskID: Long	Метод для нахождения задания на курсе
updateTask	public	Метод	newTask: TAskEntity, taskID: Long	Метод для обновления задания
deleteTask	public	Метод	taskID: Long	Метод для удаления задания
findTasks	public	Метод	courseID: Integer, start: Long	Метод для получения списка заданий
createTask	private	Метод	taskEntity: TaskEntity	Метод для создания задания
findAnswerTasks	private	Метод	taskID: Long, state_answer: Boolean, form: String	Метод для нахождения списка ответов на задание
createAnswer	private	Метод	taskID: Long, userTaskEntity: UserTaskEntity	Метод для создания ответа на задание

updateUserTask	private	Метод	newUserTask: UserTaskEntity	Метод для обновления записи UserTask в таблице.
Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
taskDao	private	TaskDao	get, set	Репозиторий для работы заданиями
userTaskDao	private	UserTaskDao	get, set	Репозиторий для работы с ответами на задание пользователей

Таблица 5.10

Описание методов и свойств интерфейса TaskDao.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
deleteTaskById	public	Метод	id: Long	Метод для удаления задание в БД по его ID
findAllByCourseIDAndIdGreaterThaEqual	public	Метод	courseID: Integer, start: Long	Метод для нахождения всех заданий по Course ID и ID больше чем указанный в параметре

Таблица 5.11

Описание методов и свойств класса TaskEntity.java

Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
id	private	Long	get, set	ID задания
form	private	String	get, set	Тип задания
title	private	String	get, set	Наименование задания
description	private	String	get, set	Описание задания
courseID	private	Integer	get, set	ID курса
createdBy	private	Long	get, set	ID пользователя, создавшего задание
updatedBy	private	Long	get, set	ID пользователя, обновившего задание
task_content	private	String	get, set	Текст задания
createdAt	private	LocalDateTime	get, set	Дата создания записи
updatedAt	private	LocalDateTime	get, set	Дата обновления записи

Таблица 5.12

Описание методов и свойств класс PostController.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
getPosts	public	Метод	courseID: Integer, start: Long	Метод для получения постов на курсе
updatePost	public	Метод	newPost: PostEntity, postID: Long	Метод для обновления поста
deletePost	public	Метод	postID: Long	Метод для удаления поста
createPost	private	Метод	postEntity: PostEntity	Метод для создания поста
Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
logger	private	Logger	get, set	Логгер.
postService	private	PostService	get, set	Сервис для работы с постами

Таблица 5.13

Описание методов и свойств класса PostService.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
findAllPosts	public	Метод	courseID: Integer, start: Long	Метод для нахождения нужных постов на курсе
updatePost	public	Метод	newPost: PostEntity, postID: Long	Метод для обновления поста
deletePost	public	Метод	postID: Long	Метод для удаления поста
createPost	private	Метод	postEntity: PostEntity	Метод для создания поста
Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
postDao	private	PostDao	get, set	Репозиторий постов.

Таблица 5.14

Описание методов и свойств интерфейса PostDao.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
deletePostById	public	Метод	id: Long	Метод для удаления поста в БД по его ID

findAllByCourseIDAndIdGreaterThaEqual	public	Метод	courseID: Integer, start: Long	Метод для нахождения всех постов по Course ID и ID больше чем указанный в параметре

Таблица 5.15

Описание методов и свойств класса PostEntity.java

Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
id	private	Long	get, set	ID поста
title	private	String	get, set	Наименование поста
content	private	String	get, set	Описание поста
courseID	private	Integer	get, set	ID курса
media_link	private	Long	get, set	Ссылка на медиа поста
doc_link	private	Long	get, set	Ссылка на файлы поста
createdAt	private	LocalDateTime	get, set	Дата создания записи
updatedAt	private	LocalDateTime	get, set	Дата обновления записи

Таблица 5.16

Описание методов и свойств класса ChatController.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
getGroupForCourse	public	Метод	groupID: Long	Метод для получения группы курса
getListMembers	public	Метод	groupID: Long	Метод для получения списка участников группы

deleteGroup	public	Метод	groupD: Long	Метод для удаления группы
getGroups	public	Метод	courseID: Integer	Метод для получения списка групп на курсе
createMessage	private	Метод	messageEntity: MessageEntity	Метод для создания сообщения
getMessageForChat	private	Метод	groupID, messageID: Long	Метод для получения сообщения в чате
deleteMessage	private	Метод	groupID, messageID: Long	Метод для удаления сообщения в чате
getMessages	private	Метод	groupID, start: Long	Метод для поучения списка сообщений в чате
deleteMember	private	Метод	groupID, userID: Long	Метод для удаления участника из группы
getMember	private	Метод	groupID, userID: Long	Метод для получения участника группы
Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
logger	private	Logger	get, set	Логгер.
chatService	private	ChatService	get, set	Сервис для работы с чатами.

Таблица 5.17

Описание методов и свойств класса ChatService.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
getGroupForCourse	public	Метод	groupID: Long	Метод для нахождения группы курса
getMembers	public	Метод	groupID: Long	Метод для получения списка участников группы

deleteGroup	public	Метод	groupD: Long	Метод для удаления группы
findAllGroups	public	Метод	courseID: Integer	Метод для получения списка групп на курсе
createMessage	private	Метод	messageEntity: MessageEntity	Метод для создания сообщения
getMessageForChat	private	Метод	groupID, messageID: Long	Метод для нахождения сообщения в чате
deleteMessage	private	Метод	groupID, messageID: Long	Метод для удаления сообщения в чате
findMessages	private	Метод	groupID, start: Long	Метод для нахождения списка сообщений в чате
deleteMember	private	Метод	groupID, userID: Long	Метод для удаления участника из группы
getMember	private	Метод	groupID, userID: Long	Метод для получения участника группы
addMember	private	Метод	groupID, userID: Long	Метод для получения участника группы
Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
chatDao	private	ChatDao	get, set	Репозиторий для работы чатами
chatMemberDao	private	ChatMemberDao	get, set	Репозиторий для работы с участниками чатов
messageDao	private	MessageDao	get, set	Репозиторий для работы с сообщениями чатов

Таблица 5.18

Описание методов и свойств интерфейса ChatDao.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
deleteGroupById	public	Метод	id: Long	Метод для удаления группы в БД по его ID
findAllByCourseID	public	Метод	courseID: Integer	Метод для нахождения всех групп по Course ID

Таблица 5.19

Описание методов и свойств класса ChatEntity.java

Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
id	private	Long	get, set	ID группы
title	private	String	get, set	Наименование группы
description	private	String	get, set	Описание группы
courseID	private	Integer	get, set	ID курса
group_avatar	private	Long	get, set	Ссылка на аватар группы
createdAt	private	LocalDateTime	get, set	Дата создания записи

Таблица 5.20

Описание методов и свойств интерфейса ChatMemberDao.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
deleteByChatIDAndUserID	public	Метод	groupID, userID: Long	Метод для удаления участника группы в БД по его ID и ID группы
findByChatIDAndUserID	public	Метод	chatID, userID: Long	Метод для нахождения участника группы в БД по его ID и ID группы
findAllMembers	public	Метод	groupID: Long	Метод для поиска всех участников группы в БД по ID группы
findMember	public	Метод	groupID, userID: Long	Метод для поиска пользователя в БД по его ID и ID группы

Таблица 5.21

Описание методов и свойств класса ChatMemberEntity.java

Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
id	private	Long	get, set	ID участника группы
chatID	private	String	get, set	ID группы
userID	private	Long	get, set	ID пользователя
updatedAt	private	LocalDateTime	get, set	Дата обновления записи

createdAt	private	LocalDateTime	get, set	Дата создания записи

Таблица 5.22

Описание методов и свойств интерфейса ProfileDao

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
getByUserID	public	Метод	id: Long	Метод для поиска профиля пользователя по UserID

Таблица 5.23

Описание методов и свойств класса ProfileEntity.java

Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
id	private	Long	get, set	ID профиля
userID	private	Long	get, set	ID пользователя
avatar	private	String	get, set	Ссылка на аватар профиля
updatedAt	private	LocalDateTime	get, set	Дата обновления записи
createdAt	private	LocalDateTime	get, set	Дата создания записи

Таблица 5.24

Описание методов и свойств интерфейса RequestDao.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
findByUserIDAndCourseID	public	Метод	userID: Long, courseID: Integer	Метод для поиска заявки пользователя по его ID и courseID

findAllByCourseIDAndApproved	public	Метод	courseID: Integer, approved: Boolean	Метод для поиска заявок пользователей по фильтрам: статус и courseID

Таблица 5.25

Описание методов и свойств класса RequestEntity.java

Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
id	private	Long	get, set	ID заявки
userID	private	Long	get, set	ID пользователя
courseID	private	Integer	get, set	ID курса
roleID	private	Integer	get, set	ID роли пользователя
approved	private	Boolean	get, set	Статус заявки
updatedAt	private	LocalDateTime	get, set	Дата обновления записи
createdAt	private	LocalDateTime	get, set	Дата создания записи

Таблица 5.26

Описание методов и свойств класса RoleEntity.java

Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
id	private	Long	get, set	ID роли пользователя
role_user	private	String	get, set	Название роли пользователя

Таблица 5.27

Описание методов и свойств интерфейса UserCourseDao.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
findByUserIDAndCourseID	public	Метод	userID: Long, courseID: Integer	Метод для поиска пользователя курса по его ID и courseID

Таблица 5.28

Описание методов и свойств класса UserCourseEntity.java

Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
id	private	Long	get, set	ID пользователя курса
userID	private	Long	get, set	ID пользователя
courseID	private	Integer	get, set	ID курса
roleID	private	Integer	get, set	ID роли пользователя

Таблица 5.29

Описание методов и свойств интерфейса UserTaskDao.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
findByTaskIDAndCourseID	public	Метод	userID: Long, taskID: Long	Метод для поиска ответа на задание пользователя по ID задания и courseID
getListOfAnswers	public	Метод	taskID: Long, status: Boolean, form: String	Метод для поиска ответов пользователей на задание по фильтрам: статус и тип задания

Таблица 5.30

Описание методов и свойств класса UserTaskEntity.java

Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
id	private	Long	get, set	ID ответа на задание
userID	private	Long	get, set	ID пользователя
answer	private	String	get, set	Наименование задания
score	private	String	get, set	Оценка ответа на задание
status	private	String	get, set	Статус ответа на задание
taskID	private	Long	get, set	ID задания
courseID	private	Integer	get, set	ID курса
createdBy	private	Long	get, set	ID пользователя, создавшего ответ
updatedBy	private	Long	get, set	ID пользователя, обновившего оценку
createdAt	private	LocalDateTime	get, set	Дата создания записи
updatedAt	private	LocalDateTime	get, set	Дата обновления записи

Таблица 5.31

Описание методов и свойств класса AppException.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
AppException	public	Конструктор	message: String	Главный класс exception в приложении. Наследуется от RuntimeException

Таблица 5.32

Описание методов и свойств класса BusinessException.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
BusinessException	public	Конструктор	message: String	Класс-наследник от appException для ошибок бизнес-логики

Таблица 5.33

Описание методов и свойств класса TechnicalException.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
TecnicalException	public	Конструктор	message: String	Класс-наследник от appException для ошибок сервера(технических)

Таблица 5.34

Описание методов и свойств класса ErrorResponse.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
ErrorResponse	public	Конструктор	String error	Класс для хранения сообщения об ошибке
Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
error	private	String	get, set	Сообщение ошибки

Таблица 5.35

Описание методов и свойств класса ExceptionMapper.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение

handleBusiness Exception	protected	Метод	e: BusinessException	Метод для обработки BusinessException
handleBusiness Exception	protected	Метод	e: TechnicalException	Метод для обработки TechnicalException
Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
log	private	Logger	get, set	Логгер

Таблица 5.36

Описание методов и свойств класса ExceptionMapper.java

Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
OBJECT_NOT_FOUND	public	String	get, set	Сообщение о не нахождении обьекта
USER_NOT_FOUND	public	String	get, set	Сообщение о не нахождении пользователя
USER_ALREADY_EXISTS	public	String	get, set	Сообщение о том, что пользователь уже существует
OBJECT_ALREADY_DELETED	public	String	get, set	Сообщение о том, что обьект уже был удален
INCORRECT_DATA	public	String	get, set	Сообщение о некорректных данных
NO_ACCESS_DUE_TO_ROLE	public	String	get, set	Сообщение о нехватке прав доступа
OBJECT_ALREADY_EXISTS	public	String	get, set	Сообщение о том, что обьект уже существует

Таблица 5.37

Описание методов и свойств класса InternalErrorResponse.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
getMessage	public	Метод		Override метод для вывода специального

				сообщения при технических ошибках

Таблица 5.38

Описание методов и свойств класса OpenApiConfig.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
openApi	public	Метод		Метод для определения bean spring API, настройки документации

Таблица 5.39

Описание методов и свойств класса JWTAuthFilter.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
doFilterInternal	protected	Метод	HttpServletRequest :request, HttpServletResponse :response, FilterChain : filterChain	Метод для декодирования JWT токена и получения информации и пользователе
Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
userDao	private	UserDao	get, set	Репозиторий пользователей
userService	private	UserService	get, set	Сервис для работы с пользователями

Таблица 5.40

Описание методов и свойств класса SecurityConfig.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
configure	protected	Метод	http: HttpSecurity	Метод для настройки конфигурации безопасности
Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение

jwtFilter	private	JWTAuthFilter	get, set	Класс для декодирования JWT токена

Таблица 5.41

Описание методов и свойств класса UserAndRole.java

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
UserAndRole	public	Конструктор	username, email: String id: Long	Класс для описания информации о пользователе и его ролях
Свойства				
Имя	Модификатор доступа	Тип	Доступ	Назначение
username	private	String	get, set	ФИО пользователя
email	private	String	get, set	Email пользователя
id	private	Long	get, set	ID пользователя

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц в докум.)	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
Изм.	Измененных	Замененных	Новых	Аннулирован					