Министерство науки и высшего образования
Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение
высшего образования
Национальный исследовательский ядерный университет
«МИФИ»

Институт интеллектуальных кибернетических систем

Кафедра №22 «Кибернетика»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ ПО ДИСЦИПЛИНЕ
Проектирование информационных систем

НА ТЕМУ:
«Система оценки знаний и психологического тестирования студентов»

Студент М23–524 (Группа) _____ (Подпись, дата) Леонов В. В. (И. О. Фамилия)

Руководитель курсовой работы _____ (Подпись, дата) Тихомирова Д. В. (И. О. Фамилия)

Москва, 2024 г.

СОДЕРЖАНИЕ

1 Обзор системы

1.1 Формулировка задания

Спроектировать и реализовать модель базы данных, а также систему оценки знаний и психологического состояния студентов. Система должна предоставлять гибкие широкие возможности по созданию новых, модификации существующих анкет, проведению и анализу тестовых контролей (как академических, так и психологических) среди студентов образовательных организаций.

1.2 Назначение и основные цели создания системы

Разработка комплексной системы, которая предназначена для обеспечения возможности учебным учреждениям более точно и эффективно оценивать знания студентов и предоставлять психологическую поддержку для студентов, включает в себя выполнение следующих целей:

1. *Оценка академических знаний.* Разработка системы позволит учебным учреждениям проводить более объективную и автоматизированную оценку знаний студентов. Это включает создание алгоритмов для автоматической проверки заданий, создание тестовых форматов и критериев оценки, а также интеграцию с учебными программами.

2. *Оценка учебного процесса.* Разработка системы предоставит возможность детальной оценки составляющих единиц учебного процесса и выявления проблемных участков.

3. *Оценка психологического состояния.* Разработка системы психологического тестирования поможет выявить психологическое состояние студентов, их потребности и стрессы. Это позволит учебным учреждениям предоставлять более целенаправленную психологическую поддержку студентам, помимо этого оптимизировать процесс получения знаний.

4. *Улучшение образовательного процесса.* Разработаная комплексная система сделает возможным улучшение образовательного процесса через более точную оценку знаний и более эффективную психологическую

поддержку, что способствует повышению качества образования и удовлетворенности студентов.

1.3 Основные функциональные возможности системы

Для разрабатываемой системы в рамках поставленных задачи и целей необходимо реализовать следующие функциональные возможности:

1. Наличие личного кабинета у каждого пользователя системы.
2. Создание и редактирование опросов ответственным лицом.
3. Вариативность форматов опросов и конкретных типов вопросов внутри опросов.
4. Проведение опросов среди целевой группы лиц.
5. Автоматизированная система проверки и оценки тестовых работ.
6. Генерация общих и детальных отчетов о результатах проведенных мероприятий с гибкой настройкой фильтрации.

1.4 Конкретизация предметной области

Важным этапом проектирования информационных систем является конкретизация предметной области, а именно этап выделения сущностей, служащих для описания всех функционирующих и взаимодействующих объектов. Для начала необходимо ответить на вопрос: «Какие пользователи будут реализованы в системе?» (более подробно о них в следующем разделе):

1. Администратор.
2. Модератор.
3. Студент.
4. Преподаватель.
5. Психолог.

Следующим шагом выделим основные структурные сущности, характерные для учебного учреждения:

1. *Документы* персонала являются важной составляющей базы данных для образовательной организации и содержат следующую информацию: тип документа и его содержимое.

2. *Программы обучения* описывают направления, по которым осуществляется подготовка и содержат следующую информацию: название, код специальности по соответствующему приказу Министерства науки и высшего образования РФ, форма обучения и ученая степень.

3. *Учебные группы* содержат следующую информацию: название группы, год и семестр, сведения о кураторе и старосте.

4. *Учебные дисциплины* содержат следующую информацию: название, форма контроля, сведения о группе и преподавателе.

Теперь сформулируем сущности для проведения тестирований:

1. Сущности для прохождения тестов:

   - *общая информация о тесте*, включающая его название, тип теста, тип оценивания, сроки прохождения, продолжительность, описание, сведения об авторе;
   - *детальная информация по вопросам*, включающая порядковый номер, описание, тип вопроса, варианты ответы и ключи для проверки (в случае если они подразумеваются).

2. Сущности о прохождении тестов:

   - *общие результаты*, включающие информацию об оценке, о проверяющем и его комментарии;
   - *детальные данные* по ответу на каждый вопрос, а также опциональные файлы от респондентов при необходимости.

1.5 Пользователи системы

В рамках данной системы оценки знаний и психологического состояния предлагается выделить несколько ролей пользователей, которые можно разделить на две большие группы: системный персонал и персонал университета.

К системному персоналу относятся:

1. ***Модератор***, обладающий возможностью полной манипуляции (добавление, изменение, удаление) всеми данными системы, за исключением данных и ролями пользователей;

2. ***Администратор***, обладающий возможностьями роли «Модератор», а также имеющий достаточный уровень доверия для манипуляции с данными и ролями пользователей системы.

Среди персонала университета разумно разделить пользователей по их выполняемым задачам в рамках работы в образовательной организации:

1. ***Студенты*** — описываются ФИО, датой рождения, контактной информацией, номером студенческого билета и личного дела;

2. ***Преподаватели*** — описываются ФИО, датой рождения, контактной информацией и учебным отделением;

3. ***Психологи*** — описываются ФИО, датой рождения, контактной информацией, номером лицензии, персональным сайтом и областью экспертизы.

Следует отметить, что возможны ситуации множественных ролей для одного пользователя (например, мистер К. может обучаться по программе аспирантуры и быть студентом, и в то же время являться преподавателем для обучающихся младших курсов).

1.6 Сроки хранения информации

Сроки хранения информации в системе оценки знаний и психологического тестирования студентов могут варьироваться в зависимости от типа информации, законодательных требований и политики организации. Выделяются два типа класса данных *по длительности хранения*: *временные* и

постоянные; два типа *по необходимости доступа*: *актуальные* и *архивные*. На их основе категоризируются различные данные:

1. **Результаты учебных тестов**.
   По длительности хранения: постоянные.
   По необходимости доступа: на протяжении периода обучения являются актуальными, после завершения обучения — архивными.

2. **Результаты психологических**.
   По длительности хранения: постоянные.
   По необходимости доступа: на протяжении периода обучения + 5 лет после завершения обучения являются актуальными, по окончании указанного ранее срока — архивными.

3. **Учебные/психологичесие тесты**.
   По длительности хранения: постоянные.
   По необходимости доступа: на протяжении ведения курса/проводимого исследования являются актуальными, после завершения учебной программы/проводимого исследования — архивными.

4. **Учетные данные**.
   По длительности хранения: постоянные.
   По необходимости доступа: на протяжении периода работы сотрудника/учебы студента являются актуальными, после завершения — архивными.

5. **Данные, описывающие учебное учреждение**:
   По длительности хранения: постоянные.
   По необходимости доступа: всегда актуальные.

1.7 События изменяющие состояние системы

События можно разделить на две группы: изменяющие состояние базы данных регулярно и нерегулярно. Регулярные изменения состояния базы данных, которые происходят с определенной периодичностью, описывают следующие события:

1. При наступлении нового учебного года требуется обновить данные о текущих студентах, перевести их на следующий курс, если они соответствуют необходимым успеваемости, а также внести информацию о новых студентах, включая первокурсников, в систему.

2. При наступлении нового семестра обучения необходимо провести обновление информации о текущих студентах в базе данных и загрузить новое расписание, а также создать новые записи о успеваемости.

3. Оценка знаний студентов включает в себя регулярное проведение контрольных мероприятий. Студенты всех курсов и направлений часто должны проходить контрольные точки несколько раз в семестр и получать оценку за каждый курс во время зачетов или экзаменов.

Далее представлены нерегулярные события:

1. Изменения в составе кадров, такие как увольнение сотрудника, требует обновления сущностей. Это включает в себя метку бывшего сотрудника, а также перемещение его в архив.

2. Изменения в составе учебных групп, такие как отчисление студента или переход студента с одного направления обучения на другое, также требуют обновления информации в базе данных только относительно студентов.

3. Создание психологических тестов и оценка их результатов предполагает, что эксперты должны внести информацию о тестах, включая список вопросов, возможные варианты ответов и описание.

4. Создание запросов на формирование отчетов предполагает, что по требованию сотрудника на основании выбранных критериев будет получен новый либо уже ранее сформированный отчет.

Таким образом, описанные категории включают события, при которых неизбежно произойдет изменение состояния данных в системе. В результате таких событий информация будет либо добавлена, либо изменена, либо перемещена в архив.

1.8 Модель жизненного цикла системы

Жизненный цикл программного обеспечения охватывает стадии, начиная с возникновения концепции идеи и завершаясь развертыванием в коде, интеграцией в бизнес-процессы и последующей поддержкой. Модели жизненного цикла ПО представляют собой обобщенный план этапов, определяющий, как происходит развитие программного продукта.

Существует значительное разнообразие в том, как проходят этапы жизненного цикла в процессе создания и внедрения продукта на рынок. Для каждого конкретного продукта этот процесс уникален, но чтобы обеспечить более системное управление и контроль, были разработаны модели жизненного цикла ПО. Они представляют собой упрощенное и обобщенное представление того, как происходит развитие продукта, и служат ориентиром для разработчиков и команд, задействованных в проекте.

Существует несколько моделей жизненного цикла системы, каждая из которых представляет собой процесс разработки и сопровождения информационной системы. далее представлены основные из них:

1. **Каскадная модель** (*Waterfall Model*). Это структурированная модель, в которой разработка системы происходит последовательно в определенных фазах: определение требований, проектирование, реализация, тестирование, внедрение и сопровождение. Каждая фаза зависит от успешного завершения предыдущей.

2. **Инкрементная модель** (*Incremental Model*). В этой модели система разрабатывается и поставляется частями или инкрементами. Каждый инкремент представляет собой функционально завершенный набор возможностей, который постепенно добавляется к системе. Эта модель позволяет быстрее получить базовую функциональность и вносить корректировки на основе обратной связи.

3. **Спиральная модель** (*Spiral Model*). Эта модель сочетает в себе элементы классической последовательной разработки с элементами итеративного и инкрементального процесса. Основное преимущество — учет рисков и возможность коррекции на ранних стадиях.

4. **Прототипирование** (*Prototyping*). В этой модели разработчики созда-

ют прототип системы, который затем уточняется на основе обратной связи пользователей. Прототип позволяет пользователям лучше понять, что им нужно от системы, и разработчикам — лучше понять требования.

5. **Итеративная модель** (*Iterative Model*). Эта модель предполагает многократное повторение процесса разработки, с каждой итерацией добавляются новые функциональности и корректируются существующие. Итерации могут быть короткими и иметь четкую цель.

6. **V-модель** (*V-Model*): Эта модель связывает каждую фазу разработки с соответствующей фазой тестирования. Это позволяет обеспечить более строгий контроль над качеством, но процесс более формален и менее гибок.

7. **Agile-модель** (*Agile Model*). Группа методологий, включая Scrum, Kanban и другие, подчеркивает гибкость и итеративность в разработке системы, акцентируя внимание на вовлечении заказчика и постоянном взаимодействии в процессе разработки.

В таблице 1.1 представлена сравнительная характеристика рассмотренных моделей ЖЦ.

Таким образом, на основе проведенного анализа Agile-модель является наиболее предпочтительной для решаемой задачи ввиду высокой гибкости и отсутсвия существенных недостатков.

Таблица 1.1 – Сравнительная таблица рассмотренных моделей ЖЦ

Модель	**Преимущества**	**Недостатки**
Каскадная	Проста в понимании и управлении Документация создается на каждой стадии	Малая гибкость, тяжело адаптировать к изменениям Длительные сроки разработки и отсутствие быстрого показателя успеха
Инкрементная	Быстрое появление базовой функциональности Возможность корректировки итераций на основе обратной связи	Требует четкого планирования и управления версиями Могут возникнуть проблемы с интеграцией и координацией
Спиральная	Акцент на управлении рисками и контроле качества Возможность адаптации к изменяющимся требованиям	Требует значительных ресурсов и времени Сложность управления процессом из-за множества итераций
Прототипирование	Возможность лучшего понимания требований пользователей Более быстрая разработка и исправление ошибок	Риск создания прототипа, который не перейдет в полноценную систему
Итеративная	Повышенная гибкость и возможность коррекции Улучшенное управление процессом и ресурсами	Требует активного участия заказчика и пользователя
V-модель	Четкая структура, гарантирующая высокое качество Возможность раннего выявления дефектов	Малая гибкость, сложность внесения изменений
Agile-модель	Высокая гибкость, способность адаптироваться к изменениям Вовлечение пользователя и активное обратное взаимодействие	Может потребовать большей дисциплины и участия заказчика

1.9 Выбор инструментальных средств проектирования

Для проектирования информационных систем используются различные инструментальные средства, которые помогают анализировать, проектировать и реализовывать систему. Предлагается применение следующих инструментов по задачам:

1. Среда для разработки — Visual Studio Code.
2. Язык программирования — Python 3.
3. Версионирование проекта — Git.
4. Средство написания документации — Latex.
5. Проектирование баз данных — ERWin.
6. Построение диаграм и чартов — Draw.io.
7. Управление проектом — Jira.

Совместное использование выбранных инструментальных средств обеспечит актуализацию документации к проектируемой системе и обеспечит мониторинг процесса внесения изменений как в документацию, описывающую разработку, так и в исходные коды программ для создания таблиц в базе данных, их модификации и заполнения. Это позволит легче управлять и отслеживать изменения в разработке, что способствует более эффективному и надежному проектированию и сопровождению информационной системы.

2 Требования к системе

2.1 Функциональные требования

Функциональные требования являются важной частью спецификации приложения и определяют, какие функции и возможности должны быть реализованы в приложении. Деление требований на пользовательские и требования домена (подсистема тестирования) может помочь структурировать их для лучшего понимания и управления.

Функциональные требования представляются следующим образом:

1. Управление данными пользователей:
    - регистрация новых студентов, преподавателей и психологов;
    - аутентификация и авторизация пользователей;
    - изменение данных учетных записей;
    - назначение ролей пользователей.
2. Управление документами пользователей:
    - загрузка новых документов;
    - удаление старых документов;
    - редактирование существующих документов.
3. Управление учебными программами:
    - добавление новых учебных программ.
4. Управление учебными дисциплинами:
    - добавление новых учебных дисциплин;
    - добаление учебной дисциплины в учебную программу;
    - назначение преподавателя.
5. Управление учебными группами:
    - создание новой учебной группы;
    - назначение куратора группы;

- назначение старосты группы;
- определение списка группы.

6. Управление тестами:
   - создание тестов различных видов;
   - возможность реализации различных видов вопросов;
   - прохождение тестирования с возможностью прикрепления дополнительных приложений;
   - определение сроков и ограничений по времени выполнения;
   - назначение целевой группы;
   - автоматическая и ручная проверка тестов;
   - наличие детального просмотра результатов.
7. Отчеты и аналитика:
   - генерация отчетов об успеваемости студентов;
   - генерация отчетов о психологическом состоянии респондентов;
   - аналитика по результатам тестирования.
8. Уведомления:
   - система уведомлений о предстоящих тестах и результатах проверки;
   - наличие сообщений об изменении учебного плана.
9. Журналирование и логирование действий пользователей, а также система мониторинга нагрузки и активности.

2.1.1 Диаграммы вариантов использования

На рисунках 2.1–2.2 приведены основные варианты использования разрабатываемой системы.

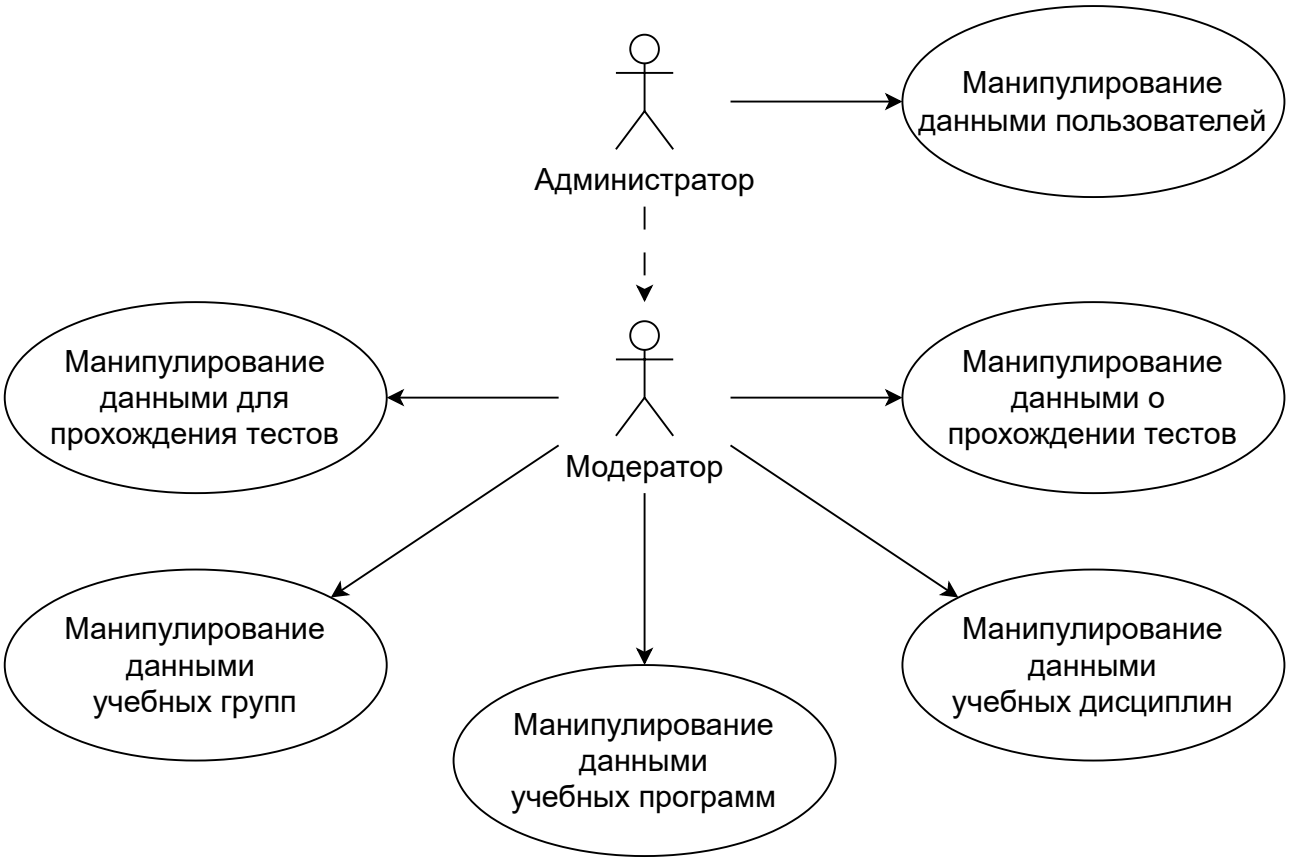


Рисунок 2.1 – Диаграмма вариантов использования (часть 1)

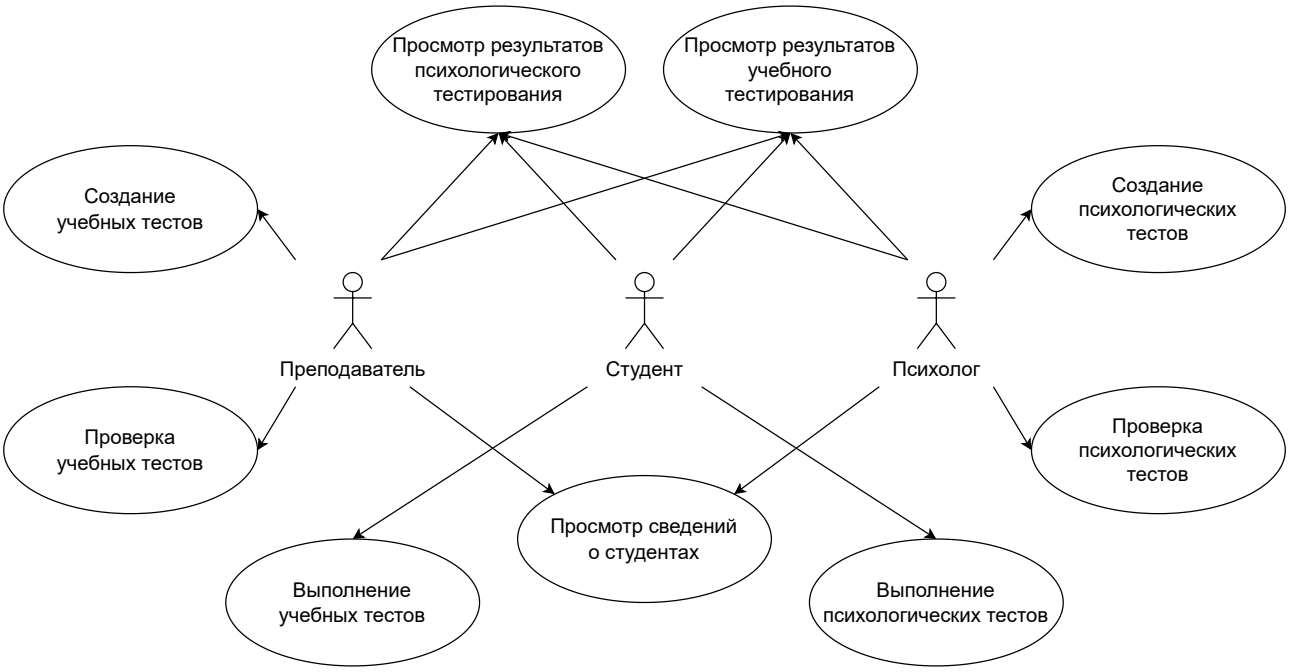


Рисунок 2.2 – Диаграмма вариантов использования (часть 2)

2.1.2 Описание вариантов использования

В приведенных ранее диаграммах вариантов использования, описанных на рисунках 2.1–2.2 под манипуляцией данными подразумеваются классические в терминах БД CRUD-операции:

1. Create — создание сущностей.
2. Read — создание сущностей.
3. Update — обновление (редактирование) сущностей.
4. Delete — удаление сущностей.

2.1.3 Дополнительные функциональные требования

Дополнительные функциональные требования не были предложены, поскольку первоначальные требования для проектируемой информационной системы были изложены весьма исчерпывающе. Это стало возможным благодаря ясному пониманию целей и задач системы при формулировании исходных требований.

2.2 Требования к экранным формам

Общие требования к интерфейсу приложения включают в себя спецификации и ожидания по дизайну, внешнему виду и его поведению. Эти требования помогают обеспечить удовлетворение потребностей пользователей, создать приятное взаимодействие и улучшить общий опыт:

1. Поддержка мультиязычности интерфейса, в том числе поддержка разных форматов дат, валют и других локальных особенностей.
2. Поддержка персонализации: наличие как светлой, так и темной темы, а также режима цветовой слепоты.
3. Соответствие единству стилистической схемы, шрифтов, иконок и досок настроения.
4. Наличие плавных анимаций для компонентов интерфейса и надежная обратная связь при взаимодействии с пользователем.

5. Адаптивность под различные платформы (Desktop, Mobile) с корректным масштабированием.

6. Поддержка широкоформатных устройств (вплоть до 32:9), а также устройств с повышенным разрешением (вплоть до 8K).

7. Интуитивная система навигации со всплывающими подсказками.

Необходимо реализовать следующие экранные формы:

1. **Форма регистрации/авторизации** должна обеспечивать функционал авторизации при помощи логина и пароля, создания новых пользователей, восстановления пароля.

2. **Форма личного кабинета студента** должна обеспечивать функционал просмотра соответствующей информации пользователя, сведений о программе обучения, учебной группе, текущих дисциплинах и тестах о них;

3. **Форма личного кабинета преподавателя/психолога** должна обеспечивать функционал просмотра соответствующей информации пользователя, сведения о последних завершенных тестах и тестах ожидающих проверки;

4. **Форма личного кабинета модератора/администратора** должна обеспечивать функционал редактирования сведений образовательной организации (учебные группы, учебные дисциплины, учебные группы и кадровый состав);

5. **Форма создания теста** должна обеспечивать функционал гибкой генерации вопросов различных видов, использования сведений из предыдущих тестов, а также выбора целевой группы для тестирования.

2.3 Нефункциональные требования

2.3.1 Общесистемные требования

Разрабатываемое приложение должно быть полным по следующим критериям:

1. Возможность масштабироваться горизонтально и вертикально, что позволит увеличивать производительность с ростом нагрузки.

2. Устойчивость к высоким нагрузкам — обработка большого количества запросов и операций без серьезных сбоев или деградации производительности.

3. Кэширование — применение кеширования данных на стороне клиента, чтобы уменьшить нагрузку на сервер и сократить время отклика.

4. Асинхронные операции и многозадачность для эффективной обработки множества запросов одновременно.

5. Поддержка всех основных операционных систем: Windows, Linux, MacOS, Android и IOS.

6. Наличие режима функционирования с потреблением ограниченного числа ресурсов, а также энергоэффективного режима для мобильных устройств.

2.3.2 Контроль входной информации

Контроль входной информации — это процесс проверки, фильтрации и обработки данных, которые поступают от пользователя или других источников на стороне клиента перед их передачей на сервер. Это важный аспект безопасности и надежности клиент-серверных приложений. Необходимо обеспечить следующие меры:

1. Валидация — верификация данных, вводимых пользователями, на соответствие ожидаемым форматам и значениям.

2. Санитизация данных — фильтрация входных данных, чтобы удалить или экранировать потенциально опасные символы/скрипты.

3. Обработка ошибок — обеспечение обработки ошибок и некорректных данных, чтобы предотвратить сбои приложения из-за неправильных данных.

4. Защита от межсайтовой подделки запросов — реализация токенов CSRF, чтобы предотвратить отправку злоумышленниками фальшивых запросов от имени пользователя.

5. Все данные на сервер передаются в зашифрованном виде.

2.3.3 Контроль выходной информации

Контроль выходной информации важен для обеспечения безопасности и конфиденциальности данных, которые передаются пользователям или другим системам. Следует реализовать следующие методы и практики контроля выходной информации:

1. Экранирование данных — перед выводом данных на экран или передачей данных на клиентскую сторону, необходимо удостовериться, что все потенциально опасные символы экранированы. Это предотвратит атаки типа XSS.

2. Защита от атак CSRF — при передаче данных на клиентскую сторону, необходимо удостовериться, что выполняется защита от атак CSRF, чтобы предотвратить изменение данных без согласия пользователя.

3. Контроль доступа к API — при взаимодействии с внешними API, необходимо удостовериться, что доступ к API ограничен и проверяется на наличие соответствующих разрешений.

4. Управление кешированием — при кешировании данных на стороне клиента, необходимо удостовериться, что данные в кеше обновляются при изменении на сервере и не сохраняют конфиденциальную информацию.

5. Логирование безопасности — ведение журнала событий безопасности, чтобы отслеживать все события, связанные с выходной информацией, и быстро реагировать на возможные инциденты.

3 Концептуально-информационная модель предметной области

Концептуальная модель предметной области — описывает основные сущности, связи между ними и их взаимодействие в рамках предметной области или системы. Эта модель не зависит от конкретной реализации и технических деталей, а сконцентрирована на понимании сущности и взаимосвязей между ними. Концептуальная модель предметной области служит основой для дальнейшего проектирования и разработки системы или базы данных.

3.1 ER-диаграмма модели

Для описания КМПО используется ER-диаграмма в нотации Чена, представленная на рисунке 3.1.

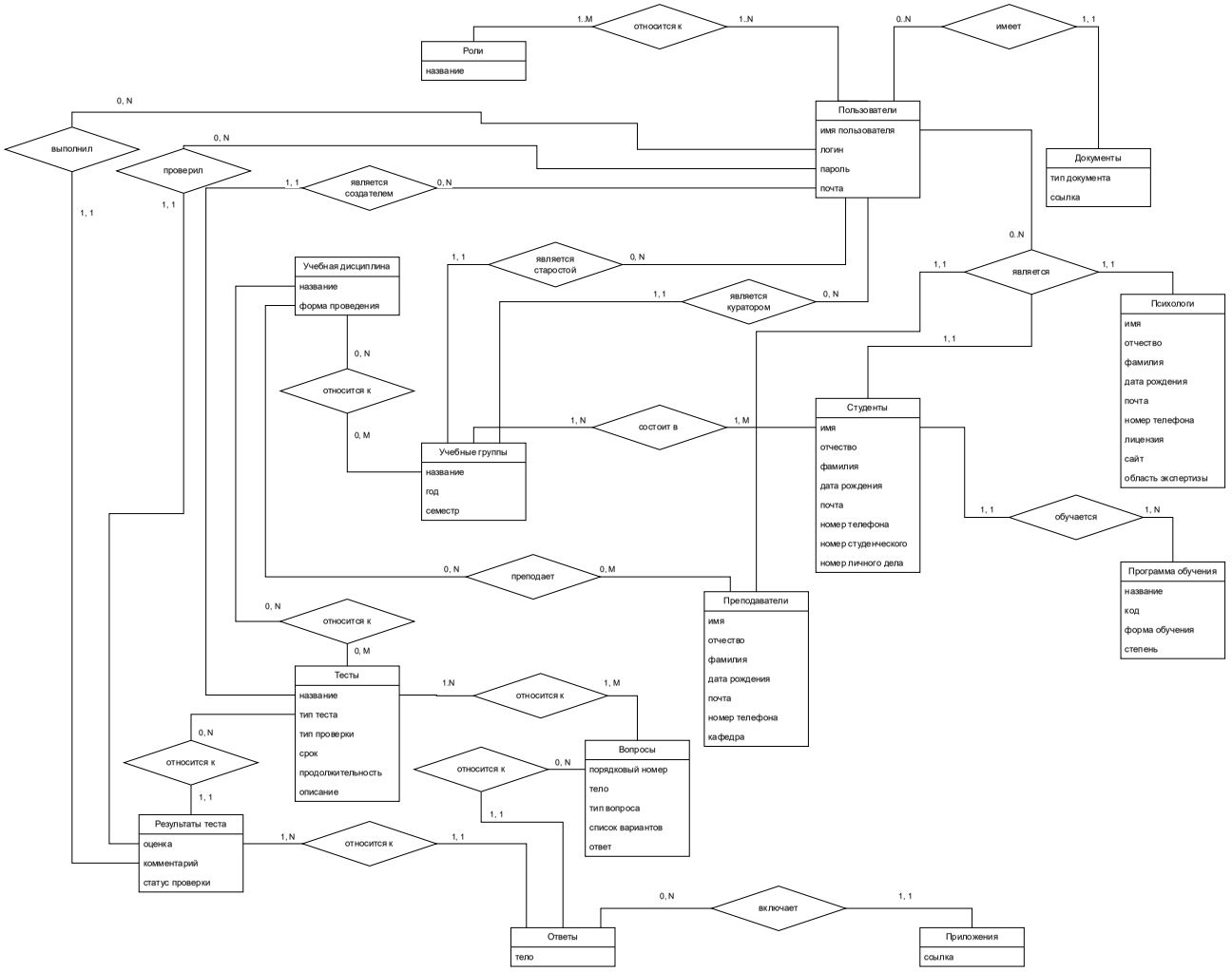


Рисунок 3.1 – Концептуальная модель предметной области

3.2 Основные запросы к базе данных (на естественном языке)

В рамках разрабатываемой системы предлагаются следующие типовые запросы:

1. Получить количество студентов, обучающихся в группах по заданной учебной программе.
2. Получить список документов заданного пользователя.
3. Получить список тестов, за которые ответственнен преподаватель и по которым есть непроверенные работы.
4. Получить контактные данные психолога.
5. Получить приложение, которые прикрепил пользователь к заданному ответу.

3.3 Оценка мощностных характеристик сущностей и связей

По Федеральному закону «Об образовании в Российской Федерации» среди высших учебных заведений про различным признакам классифицируются три вида учреждений:

- *академии* (до 2000 человек);
- *институты* (до 8000 человек);
- *университеты* (до 30000 человек).

Среди прочих признаков, размер числа обучающихся студентов является одним из критериев сравнения. Согласно данным мониторинга, проводимого Министерства образования и науки Российской Федерации среди ВУЗов средняя численность составляет 6000 человек. Ранее описывалось наличие 5 видов ролей. Как правило, учебные группы состоят из 15-20 человек. Соотношение между преподавателями и студентами приблизительно 1:20. Психологи являются представителями сторонних организаций, поэтому их число не

существенно для общей системы (до 20-30 человек). Ролью администратор обладают разработчики и технические сотрудники университета (до 50 человек), ролью модератор обладают сотрудники отдела кадров и приемной комиссии (до 50 человек). По системе кодификации направлений получения высшего образования РФ определено 300 позиций, в среднем в институте доступно до 80 направлений, в рамках каждой из которых за время обучения изучается 40 дисциплин (которых всего порядка 200-300). Основными документами для каждого из пользователей являются паспорт, СНИЛС и документ о об образовании. Для осуществления должного контроля за знаниями следует проводить в течение семестра как минимум 2 раза учебные тесты (20 вопросов, 5 из которых требуют развернутого ответа), для оценки психологического состояния — 1 раз (50 вопросов). Таким образом, на основании приведеной выше информации была произведена оценка мощностных характеристик сущностей и связей, которая оформлена в виде таблиц 3.1–3.2.

Таблица 3.1 – Таблица мощностей множеств (часть 1)

Сущность/Связь	**Минимум**	**Среднее**	**Максимум**
Пользователи	1100	6400	32000
Преподаватели	50	300	1500
Студенты	1000	6000	30000
Психологи	10	20	30
Модераторы	10	30	50
Администраторы	10	30	50
Роли	5	5	5
Учебные группы	50	300	1500
Программы обучения	40	80	120
Документы	3300	19200	96000
Учебные дисциплины	100	200	300
Тесты	500	1000	2000
Вопросы	10000	30000	80000
Результаты теста	10000	60000	300000
Ответы	200000	1800000	9000000
Приложения	2000	18000	90000
Ответ включает Приложение	2000	18000	90000
Пользователь относится к роли	1100	6400	32000

Таблица 3.2 – Таблица мощностей множеств (часть 2)

Сущность/Связь	**Минимум**	**Среднее**	**Максимум**
Пользователь имеет документы	3300	19200	96000
Ответ относится к результату теста	200000	1800000	9000000
Ответ относится к вопросу	200000	1800000	9000000
Результат теста относится к тесту	10000	60000	300000
Вопрос относится к тесту	20000	60000	160000
Тест относится к учебной дисциплине	1200	2500	3500
Преподаватель преподает учебную дисциплину	250	1500	7500
Студент обучается по программе обучения	1000	6000	30000
Пользователь является студентом	1000	6000	30000
Пользователь является преподавателем	50	300	1500
Пользователь является психологом	10	20	30
Студент состоит в учебной группе	1000	6000	30000
Пользователь является куратором группы	50	300	1500
Пользователь является старостой группы	50	300	1500
Учебная группа относится к учебной дисциплине	200	1200	7000
Пользователь является создателем теста	500	1000	2000
Результат теста выполнил пользователь	10000	60000	300000
Результат теста проверил пользователь	10000	60000	300000

4 Концептуальное проектирование

4.1 Принятые проектные соглашения

При переходе от КМПО к КМПД для каждой сущности потребовалось определить дополнительно суррогатные ключи.

4.2 Обоснование выбора модели базы данных

Модель базы данных — это тип модели данных, которая определяет логическую структуру базы данных и в корне определяет, каким образом данные могут храниться, организовываться и обрабатываться. Рассмотрим основные модели.

Иерархическая модель

В иерархической модели информация организована в виде древовидной структуры. Каждая запись имеет одного «родителя» и несколько потомков.

Такие модели данных графически могут быть представлены в виде перевернутого дерева. Оно состоит состоящее из объектов, каждый из которых имеет уровень. Корень дерево — первый уровень, его потомки второй и так далее.

Основной недостаток иерархической модели данных — невозможно реализовать отношение "many-to-many связь, при которой у потомка существует несколько родителей.

В качестве примера такой модели можно привести каталог в операционной системе Windows.

Сетевая модель

Сетевая модель — это структура, у которой любой элемент может быть связан с любым другим элементом.

Сетевая модель описывается как иерархическая, но в отличие от последней лишена недостатков, связанных с невозможностью реализовать "many-to-many"связь. Разница между сетевой и иерархической заключается в том, что в сетевой модели у потомка может быть несколько предков, когда у иерархической только один.

Основной недостаток — жёсткость задаваемых структур и сложность изменения схем БД из-за реализации связей между объектами на базе физических ссылок (через указатели на объекты).

В качестве примера такой модели можно привести WWW (World Wide Web).

Реляционная модель

Данные в реляционной модели хранятся в виде таблиц и строк, таблицы могут иметь связи с другими таблицами через внешние ключи, таким образом образуя некие отношения.

Реляционные базы данных используют язык SQL. Структура таких баз данных позволяет связывать информацию из разных таблиц с помощью внешних ключей (или индексов), которые используются для уникальной идентификации любого атомарного фрагмента данных в этой таблице. Другие таблицы могут ссылаться на этот внешний ключ, чтобы создать связь между частями данных и частью, на которую указывает внешний ключ.

SQL используют универсальный язык структурированных запросов для определения и обработки данных. Это накладывает определенные ограничения: прежде чем начать обработку, данные надо разместить внутри таблиц и описать.

Нереляционная модель

Данные нереляционных баз данных не имеют общего формата. Они могут представляться в виде документов (Mongo, Tarantool), пар ключ-значение (Redis), графовых представлениях.

Динамические схемы для неструктурированных данных позволяют:

- ориентировать информацию на столбцы или документы;
- основывать ее на графике;
- организовывать в виде хранилища Key-Value;
- создавать документы без предварительного определения их структуры, использовать разный синтаксис;
- добавлять поля непосредственно в процессе обработки.

Для решения задачи будет использоваться реляционная модель данных по нескольким причинам:

- изложение данных будет осуществляться в виде таблиц;

- данные структурированные, структура нечасто измяема;
- возможность исключить дублирование, используя связь между отношениями с помощью внешних ключей;
- разделение доступа к данным от способа их физической организации.

Для хранения состояний системы, в частности, авторизации пользователей, следует использовать нереляционную модель. Особенность таких данных в том, что у них нет отношений и связей, они хранятся парами key-value. Главное требование — данная база должна отличаться быстродействием.

Для решения таких задач следует рассмотреть in-memory СУБД.

In-Memory —- это набор концепций хранения данных, когда они сохраняются в оперативной памяти приложения, а диск используется для бэкапа. В классических подходах данные хранятся на диске, а память — в кэше.

Такие СУБД основаны на нереляционной модели данных.

4.3 Используемые в системе кодификаторы

В рамках разрабатываемой системы были определены следующие кодификаторы:

Таблица 4.1 – Кодификатор «Роль пользователя»

Код	**Значение**
administrator	Администратор
moderator	Модератор
student	Студент
teacher	Преподаватель
psychologist	Психолог

Таблица 4.2 – Кодификатор «Форма обучения»

Код	**Значение**
face-to-face	Очная
distance	Заочная
hybrid	Очно-заочная

Таблица 4.3 – Кодификатор «Степень»

Код	**Значение**
bachelor	Бакалавриат
master	Магистратура
graduate	Аспирантура
specialty	Специалитет

Таблица 4.4 – Кодификатор «Семестр»

Код	**Значение**
autumn	Осенний семестр
spring	Весенний семестр

Таблица 4.5 – Кодификатор «Тип теста»

Код	**Значение**
psychological	Психологический
educational	Учебный

Таблица 4.6 – Кодификатор «Тип результата»

Код	**Значение**
auto	Авто проверка теста
manual	Ручная проверка теста

Таблица 4.7 – Кодификатор «Статус проверки»

Код	**Значение**
to-be-checked	Ожидает проверки
to-be-confirmed	Ожидает подтверждения
ready	Готово

Таблица 4.8 – Кодификатор «Вид контроля»

Код	**Значение**
pass	Зачет
credit	Оценка

4.4 Концептуальная модель базы данных

На рисунке 4.1 представлена концептуальная модель базы данных.

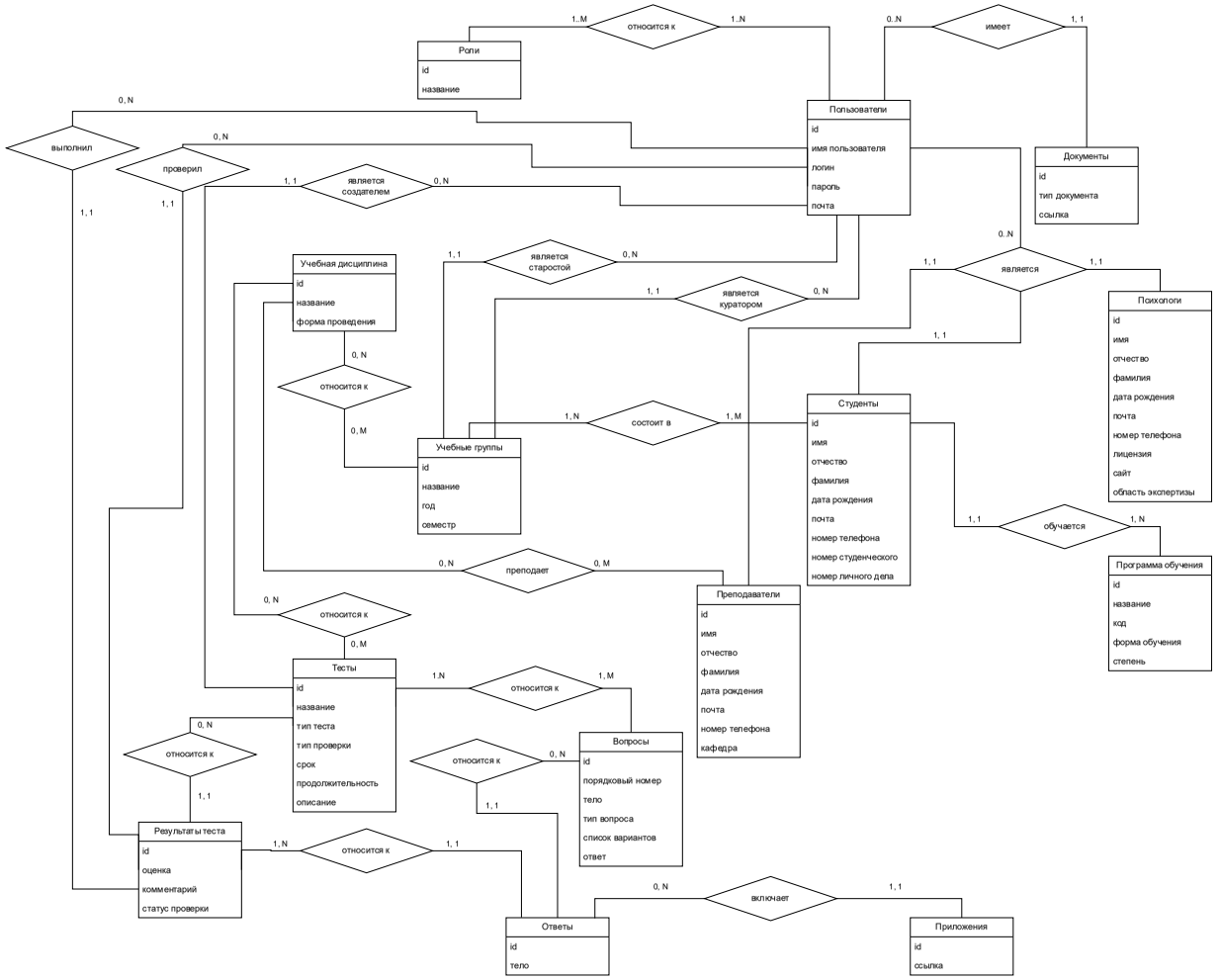


Рисунок 4.1 – Концептуальная модель базы данных

5 Логическое проектирование

5.1 ER-диаграмма базы данных

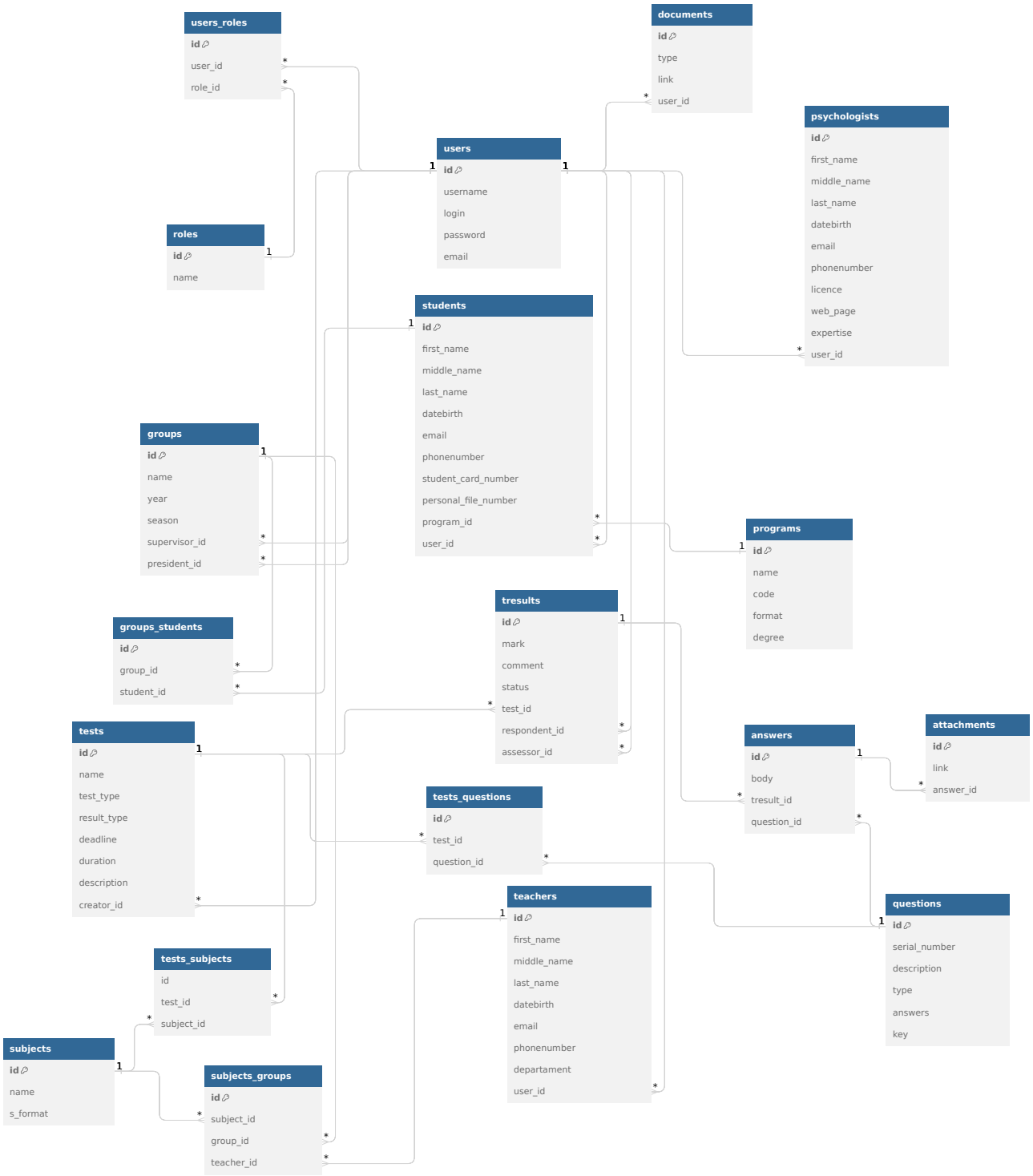


Рисунок 5.1 – ER-диаграмма базы данных

5.2 Схемы отношений базы данных

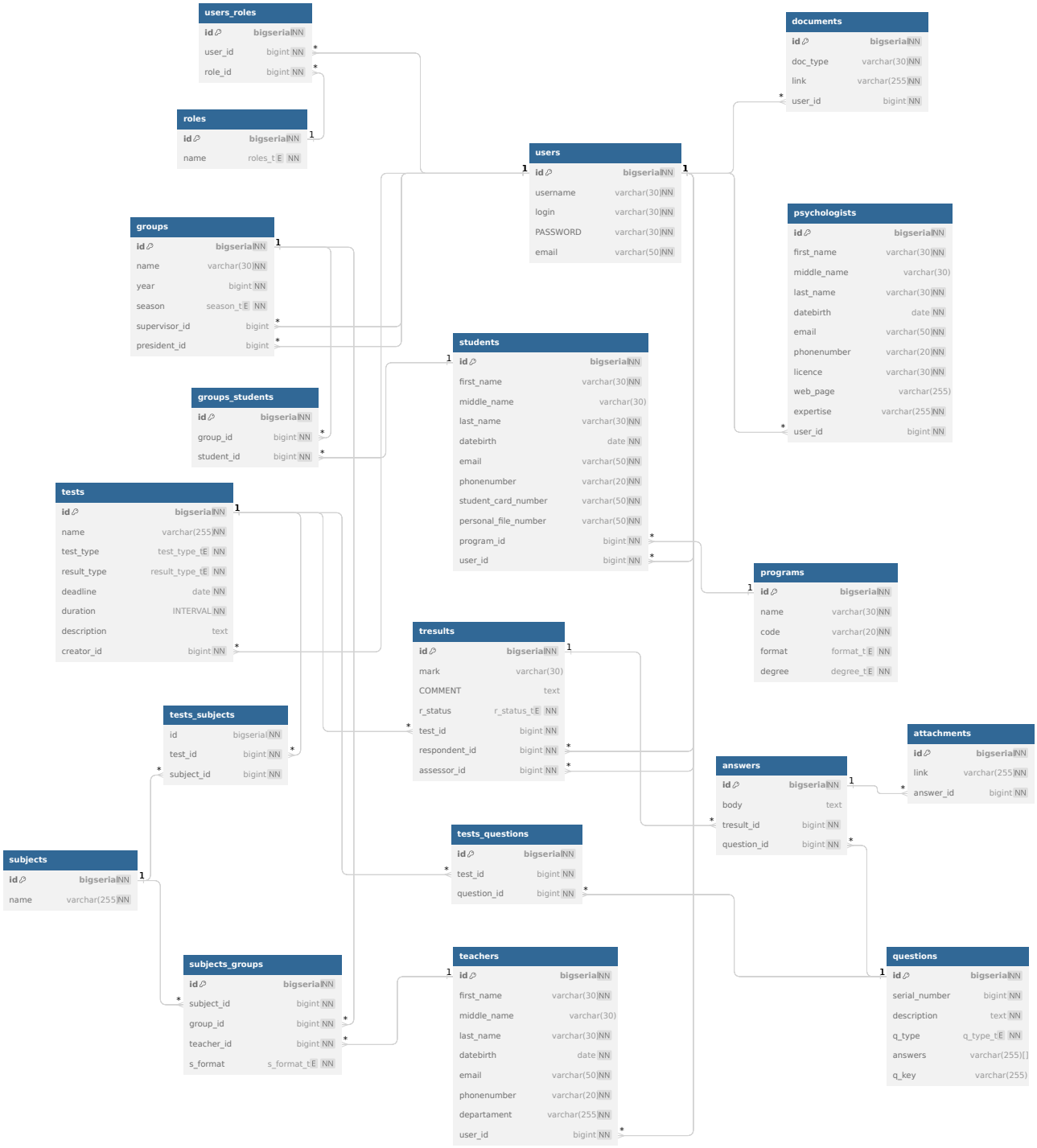


Рисунок 5.2 – Схема отношений базы данных

5.3 Схема реляционной базы данных

Таблица 5.1 – Схема реляционной базы данных

R1	users (#id, username, login, password, email)
R2	roles(#id, name)
R3	users_roles(#id, user_id, role_id)
R4	students (#id, first_name, middle_name, last_name, datebirth, email, phonenumber, student_card_number, personal_file_number, program_id, user_id)
R5	teachers (#id, first_name, middle_name, last_name, datebirth, email, phonenumber, departament, user_id)
R6	psychologists (#id, first_name, middle_name, last_name, datebirth, email, phonenumber, license, web_page, expersite, user_id)
R7	documents (#id, doc_type, link, user_id)
R8	programs (#id, name, code, format, degree)
R9	groups (#id, name, year, season, supervisor_id, president_id)
R10	groups_students (#id, group_id, student_id)
R11	tests (#id, name, test_type, result_type, deadline, duration, description, creator_id)
R12	questions (#id, serial_number, description, q_type, answers, q_key)
R13	tests_questions (#id, test_id, question_id)
R14	tresults (#id, mark, comment, r_status, test_id, respondent_id, assessor_id)
R15	answers (#id, body, tresult_id, question_id)
R16	attachments (#id, link, answer_id)
R17	subjects (#id, name)
R18	subjects_groups (#id, subject_id, group_id, teacher_id, s_format)
R19	test_subjects (#id, test_id, subject_id)

5.4 Схемы основных запросов на реляционной алгебре

Реляционная алгебра, используемая в базах данных, опирается на набор операций, применяемых к отношениям (таблицам) для получения желаемых

результатов. Основные операции реляционной алгебры включают соединение, проекцию, селекцию, объединение, разность, и пересечение. Вот как эти операции обычно обозначаются в нотации реляционной алгебры и как они обозначаются в данной работе:

- ***Селекция***. Обозначается символом σ. Это унарная операция, используемая для фильтрации строк таблицы по заданному критерию. Например, σ условие (Таблица).

- ***Проекция***. Обозначается символом π. Это унарная операция, которая выбирает определенные столбцы из таблицы. Например, π столбец1, столбец2 (Таблица).

- ***Соединение***. Обозначается символом $\bowtie$. Это бинарная операция, которая комбинирует строки двух таблиц на основе общего условия. Например, Таблица1 $\bowtie$ условие Таблица2.

- ***Пересечение***. Обозначается символом $\cap$. Эта операция возвращает строки, которые присутствуют в обеих таблицах.

- ***Объединение***. Обозначается символом $\cup$. Эта операция объединяет строки двух таблиц, исключая дубликаты.

- ***Разность***. Обозначается символом $-$. Эта операция возвращает строки, присутствующие в первой таблице, но отсутствующие во второй.

- ***Декартово произведение***. Обозначается символом $\times$. Это бинарная операция, которая комбинирует каждую строку одной таблицы с каждой строкой другой таблицы.

Получить количество студентов магистратуры, обучающихся в группах по заданной учебной программе

π groups.name, count (*) students $\bowtie$ students.id = groups_students.student_id groups_students $\bowtie$ groups.id = groups_students.id groups $\bowtie$ students.program_id = programs.id programs σ programs.name = 'Программная инженерия' and programs.format = 'master' group by group.name;

Получить список документов заданного пользователя

π documents.document_type, documents.link documents σ documents.user_id = 5;

Получить список тестов, за которые ответственнен преподаватель и по которым есть непроверенные работы

π distinct tests.name, tests.deadline, tests.description tests $\bowtie$ tests.creator_id = users.id users $\bowtie$ users.id = teachers.user_id teachers $\bowtie$ tresults.test_id = tests.id tresults σ teachers.first_name = 'Сергей' and teachers.middle_name = 'Викторович' and teachers.last_name = 'Демченко' and tresults.r_status = 'to-be-checked' order by tests.deadline;

Получить контактные данные психолога

π psychologists.first_name || ' ' || psychologists.middle_name || ' ' || psychologists.last_name, psychologists.phonenumber, psychologists.email, psychologists.web_page psychologists σ psychologists.first_name = 'Игорь' and psychologists.middle_name = 'Владимирович' and psychologists.last_name = 'Стольников';

Получить приложение, которые прикрепил пользователь к заданному ответу

π answers.id, attachments.link answers $\bowtie$ answers.id = attachments.id attachments σ answers.id = 27;

6 Физическое проектирование

6.1 Обоснование выбора конкретной СУБД

В данном подразделе буду рассмотрены популярные построчные СУБД, которые могут быть использованы для реализации хранения в разрабатываемом программном продукте.

PostgreSQL

PostgreSQL – это свободно распространяемая объектно-реляционная система управления базами данных, наиболее развитая из открытых СУБД в мире.

PostgreSQL предоставляет транзакции со свойствами атомарности, согласованности, изоляции, долговечности (ACID), автоматически обновляемые представления, материализованные представления, триггеры, внешние ключи и хранимые процедуры.

Рассматриваемая СУБД управляет параллелизмом с помощью технологии управления многоверсионным параллелизмом (англ. MVCC). Эта технология дает каждой транзакции «снимок» текущего состояния базы данных, позволяя вносить изменения, не затрагивая другие транзакции. Это в значительной степени устраняет необходимость в блокировках чтения и гарантирует, что база данных поддерживает принципы ACID.

Oracle Database

Oracle Database – объектно-реляционная система управления базами данных компании Oracle. На данный момент, рассматриваемая СУБД является самой популярной в мире.

Все транзакции Oracle Database соответствуют обладают свойствами ACID, поддерживает триггеры, внешние ключи и хранимые процедуры. Данная СУБД подходит для разнообразных рабочих нагрузок и может использоваться практически в любых задачах. Особенностью Oracle Database является быстрая работа с большими массивами данных.

Oracle Database может использовать один или более методов параллелизма. Сюда входят механизмы блокировки для гарантии монопольного использования таблицы одной транзакцией, методы временных меток, которые разрешают сериализацию транзакций и планирование транзакций на основе проверки достоверности.

MySQL

MySQL – свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle.

Рассматриваемая СУБД имеет два основных механизма хранения данных: InnoDB и myISAM. InnoDB полностью полностью совместим с принципами ACID, в отличии от myISAM. СУБД MySQL подходит для использования при разработке веб-приложений, что объясняется очень тесной интеграцией с популярными языками PHP и Perl.

Реализация параллелизма в СУБД MySQL реализовано с помощью механизма блокировок, который обеспечивает одновременный доступ к данным.

Выбор СУБД для решения задачи

Для решения задачи была выбрана СУБД PostgreSQL, потому что данная СУБД проста в развертывании, является свободно-распространяемым ПО, эффективно по быстродействию и обладает исчерпывающим функционалом.

6.2 Создание базы данных

Листинг 6.1 – Сценарий создания базы данных

```
1 createdb –h localhost –p 5432 –U postgres learning_system
```

6.3 Создание таблиц

Листинг 6.2 – Сценарий создания таблиц базы данных (часть 1)

```
CREATE TABLE users (
  id bigserial NOT NULL,
  username varchar (30) NOT NULL,
  login varchar (30) NOT NULL,
  PASSWORD varchar (30) NOT NULL,
  email varchar (50) NOT NULL,
  PRIMARY KEY (id)
);

CREATE TYPE roles_t AS enum(
  'administrator',
  'moderator',
  'student',
  'teacher',
  'psychologist'
);

CREATE TABLE roles (
  id bigserial NOT NULL,
  name roles_t NOT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE users_roles (
  id bigserial NOT NULL,
  user_id bigint NOT NULL,
  role_id bigint NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (user_id) REFERENCES users (id) ON DELETE CASCADE,
  FOREIGN KEY (role_id) REFERENCES roles (id) ON DELETE CASCADE
);

CREATE TYPE format_t AS enum('face-to-face', 'distance', 'hybrid');

CREATE TYPE degree_t AS enum('bachelor', 'master', 'graduate', 'specialty');

CREATE TABLE programs (
  id bigserial NOT NULL,
  name varchar (30) NOT NULL,
  code varchar (20) NOT NULL,
  format format_t NOT NULL,
  degree degree_t NOT NULL,
  PRIMARY KEY (id)
);
```

Листинг 6.3 – Сценарий создания таблиц базы данных (часть 2)

```
1
2  CREATE TABLE students (
3     id bigserial NOT NULL,
4     first_name varchar (30) NOT NULL,
5     middle_name varchar (30),
6     last_name varchar (30) NOT NULL,
7     datebirth date NOT NULL,
8     email varchar (50) NOT NULL,
9     phonenumber varchar (20) NOT NULL,
10    student_card_number varchar (50) NOT NULL,
11    personal_file_number varchar (50) NOT NULL,
12    program_id bigint NOT NULL,
13    user_id bigint NOT NULL,
14    PRIMARY KEY (id),
15    FOREIGN KEY (user_id) REFERENCES users (id) ON DELETE CASCADE,
16    FOREIGN KEY (program_id) REFERENCES programs (id) ON DELETE CASCADE
17 );
18
19 CREATE TABLE teachers (
20    id bigserial NOT NULL,
21    first_name varchar (30) NOT NULL,
22    middle_name varchar (30),
23    last_name varchar (30) NOT NULL,
24    datebirth date NOT NULL,
25    email varchar (50) NOT NULL,
26    phonenumber varchar (20) NOT NULL,
27    departament varchar (255) NOT NULL,
28    user_id bigint NOT NULL,
29    PRIMARY KEY (id),
30    FOREIGN KEY (user_id) REFERENCES users (id) ON DELETE CASCADE
31 );
32
33 CREATE TABLE psychologists (
34    id bigserial NOT NULL,
35    first_name varchar (30) NOT NULL,
36    middle_name varchar (30),
37    last_name varchar (30) NOT NULL,
38    datebirth date NOT NULL,
39    email varchar (50) NOT NULL,
40    phonenumber varchar (20) NOT NULL,
41    licence varchar (30) NOT NULL,
42    web_page varchar (255),
43    expertise varchar (255) NOT NULL,
44    user_id bigint NOT NULL,
45    PRIMARY KEY (id),
46    FOREIGN KEY (user_id) REFERENCES users (id) ON DELETE CASCADE
47 );
```

Листинг 6.4 – Сценарий создания таблиц базы данных (часть 3)

```
1
2  CREATE TABLE documents (
3     id bigserial NOT NULL,
4     doc_type varchar (30) NOT NULL,
5     link varchar (255) NOT NULL,
6     user_id bigint NOT NULL,
7    PRIMARY KEY (id),
8    FOREIGN KEY (user_id) REFERENCES users (id) ON DELETE CASCADE
9  );
10
11 CREATE TYPE season_t AS enum('autumn', 'spring');
12 CREATE TABLE groups (
13    id bigserial NOT NULL,
14    name varchar (30) NOT NULL,
15    year bigint NOT NULL,
16    season season_t NOT NULL,
17    supervisor_id bigint,
18    president_id bigint,
19    PRIMARY KEY (id),
20    FOREIGN KEY (supervisor_id) REFERENCES users (id) ON DELETE CASCADE,
21    FOREIGN KEY (president_id) REFERENCES users (id) ON DELETE CASCADE
22 );
23
24 CREATE TABLE groups_students (
25    id bigserial NOT NULL,
26    group_id bigint NOT NULL,
27    student_id bigint NOT NULL,
28    PRIMARY KEY (id),
29    FOREIGN KEY (group_id) REFERENCES groups (id) ON DELETE CASCADE,
30    FOREIGN KEY (student_id) REFERENCES students (id) ON DELETE CASCADE
31 );
32
33 CREATE TYPE test_type_t AS enum('psychological', 'educational');
34 CREATE TYPE result_type_t AS enum('auto', 'manual');
35 CREATE TABLE tests (
36    id bigserial NOT NULL,
37    name varchar (255) NOT NULL,
38    test_type test_type_t NOT NULL,
39    result_type result_type_t NOT NULL,
40    deadline date NOT NULL,
41    duration INTERVAL NOT NULL,
42    description text,
43    creator_id bigint NOT NULL,
44    PRIMARY KEY (id),
45    FOREIGN KEY (creator_id) REFERENCES users (id) ON DELETE CASCADE
46 );
```

Листинг 6.5 – Сценарий создания таблиц базы данных (часть 4)

```
CREATE TYPE q_type_t AS enum('single', 'multiple', 'open');
CREATE TABLE questions (
  id bigserial NOT NULL,
  serial_number bigint NOT NULL,
  description text NOT NULL,
  q_type q_type_t NOT NULL,
  answers varchar(255) [],
  q_key varchar (255),
  PRIMARY KEY (id)
);

CREATE TABLE tests_questions (
  id bigserial NOT NULL,
  test_id bigint NOT NULL,
  question_id bigint NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (test_id) REFERENCES tests (id) ON DELETE CASCADE,
  FOREIGN KEY (question_id) REFERENCES questions (id) ON DELETE CASCADE
);

CREATE TYPE r_status_t AS enum('to-be-checked', 'to-be-confirmed', 'ready');

CREATE TABLE tresults (
  id bigserial NOT NULL,
  mark varchar (30),
  COMMENT text,
  r_status r_status_t NOT NULL,
  test_id bigint NOT NULL,
  respondent_id bigint NOT NULL,
  assessor_id bigint NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (test_id) REFERENCES tests (id) ON DELETE CASCADE,
  FOREIGN KEY (respondent_id) REFERENCES users (id) ON DELETE CASCADE,
  FOREIGN KEY (assessor_id) REFERENCES users (id) ON DELETE CASCADE
);


CREATE TABLE answers (
  id bigserial NOT NULL,
  body text,
  tresult_id bigint NOT NULL,
  question_id bigint NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (tresult_id) REFERENCES tresults (id) ON DELETE CASCADE,
  FOREIGN KEY (question_id) REFERENCES questions (id) ON DELETE CASCADE
);
```

Листинг 6.6 – Сценарий создания таблиц базы данных (часть 5)

```
CREATE TABLE attachments (
  id bigserial NOT NULL,
  link varchar (255) NOT NULL,
  answer_id bigint NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (answer_id) REFERENCES answers (id) ON DELETE CASCADE
);

CREATE TABLE subjects (
  id bigserial NOT NULL,
  name varchar (255) NOT NULL,
  PRIMARY KEY (id)
);

CREATE TYPE s_format_t AS enum('pass', 'credit');

CREATE TABLE subjects_groups (
  id bigserial NOT NULL,
  subject_id bigint NOT NULL,
  group_id bigint NOT NULL,
  teacher_id bigint NOT NULL,
  s_format s_format_t NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (subject_id) REFERENCES subjects (id) ON DELETE CASCADE,
  FOREIGN KEY (group_id) REFERENCES groups (id) ON DELETE CASCADE,
  FOREIGN KEY (teacher_id) REFERENCES teachers (id) ON DELETE CASCADE
);

CREATE TABLE tests_subjects (
  id bigserial NOT NULL,
  test_id bigint NOT NULL,
  subject_id bigint NOT NULL,
  FOREIGN KEY (test_id) REFERENCES tests(id) ON DELETE CASCADE,
  FOREIGN KEY (subject_id) REFERENCES subjects (id) ON DELETE CASCADE
);
```

Следует отметить, что для приведенных ранее кодификаторов на уровне базы данных были выделены отдельные типы, которые позволят формально отобразить доступные коды.

Таким образом, база данных и все соответствующие таблицы созданы (см.рисунок 6.1) и готовы для работы.

```
postgres=# \dt
              Список отношений
 Схема  |       Имя        |   Тип   | Владелец
--------+------------------+---------+----------
 public | answers          | таблица | postgres
 public | attachments      | таблица | postgres
 public | documents        | таблица | postgres
 public | groups           | таблица | postgres
 public | groups_students  | таблица | postgres
 public | programs         | таблица | postgres
 public | psychologists    | таблица | postgres
 public | questions        | таблица | postgres
 public | roles            | таблица | postgres
 public | students         | таблица | postgres
 public | subjects         | таблица | postgres
 public | subjects_groups  | таблица | postgres
 public | teachers         | таблица | postgres
 public | tests            | таблица | postgres
 public | tests_questions  | таблица | postgres
 public | tests_subjects   | таблица | postgres
 public | tresults         | таблица | postgres
 public | users            | таблица | postgres
 public | users_roles      | таблица | postgres
(19 строк)
```

Рисунок 6.1 – Множество созданных таблиц

6.4 ETL-процессы загрузки базы данных

ETL (Extract, Transform, Load) — это процесс обработки данных, начиная с извлечения (Extract) из источников данных, затем трансформации (Transform) этих данных в соответствии с определенными правилами и, наконец, загрузки (Load) их в целевую систему или хранилище данных. Распишем каждый этап более подробно:

- ***Извлечение (Extract)***. В этом этапе данные извлекаются из различных источников, таких как базы данных, текстовые файлы, API или внешние системы. Процесс извлечения может включать в себя определение, какие данные необходимо извлечь, и применение оптимизаций для повышения производительности.

- ***Трансформация (Transform)***. После извлечения данные подвергаются трансформации, где выполняются различные операции для очистки,

стандартизации и обогащения данных. Этот этап включает в себя фильтрацию выбранных данных, преобразование форматов, объединение данных из разных источников, агрегацию, и другие манипуляции.

- ***Загрузка (Load)***. На последнем этапе происходит загрузка обработанных данных в целевую систему, которая может быть хранилищем данных, хранилищем для аналитики, базой данных или другим приемником. Этот этап включает в себя оптимизацию для эффективного хранения и доступа к данным.

Так для инициализации данных в базе реализован инструментарий на Python, который состоит из двух модулей:

- Модуль генерации случайных данных. Применяется для заполнения следующих таблиц:
  1. attachments
  2. documents
  3. groups
  4. groups_students
  5. psychologists
  6. roles
  7. students
  8. subjects_groups
  9. teachers
  10. users
  11. users_roles
- Модуль получения данных из открытых источников. Применяется для заполнения следующих таблиц:
  1. answers
  2. programs
  3. questions

4. subjects
5. tests
6. tests_questions
7. tests_subjects
8. tresults

Результатом работы данного инструмента являются csv-файлы, которые автоматизированным образом загружаются в базу данных.

6.5 Запросы в терминах SQL

Получить количество студентов магистратуры, обучающихся в группах по заданной учебной программе

```sql
SELECT
    g.name AS group_name,
    count (*) AS students_amount
FROM
    students s
    JOIN groups_students gs ON s.id = gs.student_id
    JOIN groups g ON g.id = gs.id
    JOIN programs p ON s.program_id = p.id
WHERE
    p.name = 'Программная инженерия'
    AND p.format = 'master'
GROUP BY
    g.name;
```

```
 group_name | students_amount
------------+-----------------
 M23-504    |              11
 M23-514    |              17
 M23-524    |               7
 M23-534    |              17
 M22-514    |              16
 M22-524    |              13
 M22-564    |               6
(7 строк)
```

Рисунок 6.2 – Результат запроса № 1

Получить список документов заданного пользователя

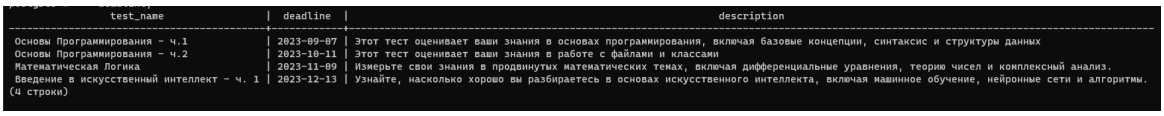
```sql
SELECT
    document_type AS document_type,
    link AS link
FROM
    documents d
WHERE
    d.user_id = 5;
```

```
          document_type          |           link
---------------------------------+--------------------------
 Паспорт                         | document_link.com/2fnka9
 СНИЛС                           | document_link.com/3kbbsf
 Аттестат                        | document_link.com/k15mx1
 Автомобильные права             | document_link.com/2z1fs1
 Военные билет                   | document_link.com/ksdsg1
 Социальная карта                | document_link.com/loasf8
 Договор о проживании в общежити | document_link.com/zxasf8
 Медицинский полис               | document_link.com/prmvxz
(8 строк)
```

Рисунок 6.3 – Результат запроса № 2

Получить список тестов, за которые ответственнен преподаватель и по которым есть непроверенные работы

```sql
SELECT
    DISTINCT t.name AS test_name,
    t.deadline AS deadline,
    t.description AS description
FROM
    tests t
    JOIN users u ON t.creator_id = u.id
    JOIN teachers tch ON u.id = t.user_id
    JOIN tresults tr ON tr.test_id = t.id
WHERE
    tch.first_name = 'Сергей'
    AND tch.middle_name = 'Викторович'
    AND tch.last_name = 'Демченко'
    AND tr.r_status = 'to-be-checked'
ORDER BY
    t.deadline;
```



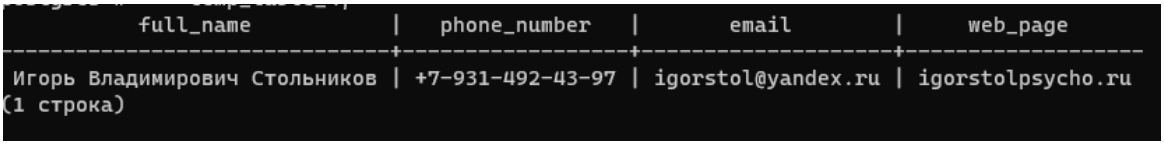
test_name	deadline	description
Основы Программирования - ч.1	2023-09-07	Этот тест оценивает ваши знания в основах программирования, включая базовые концепции, синтаксис и структуры данных
Основы Программирования - ч.2	2023-10-11	Этот тест оценивает ваши знания в работе с файлами и классами
Математическая Логика	2023-11-09	Измерьте свои знания в продвинутых математических темах, включая дифференциальные уравнения, теорию чисел и комплексный анализ.
Введение в искусственный интеллект - ч. 1	2023-12-13	Узнайте, насколько хорошо вы разбираетесь в основах искусственного интеллекта, включая машинное обучение, нейронные сети и алгоритмы.

(4 строки)

Рисунок 6.4 – Результат запроса № 3

Получить контактные данные психолога

```sql
SELECT
    p.first_name || ' ' || p.middle_name || ' ' || p.last_name AS full_name,
    p.phonenumber AS phone_number,
    p.email AS email,
    p.web_page AS web_page
FROM
    psychologists p
WHERE
    p.first_name = 'Игорь'
    AND p.middle_name = 'Владимирович'
    AND p.last_name = 'Стольников';
```



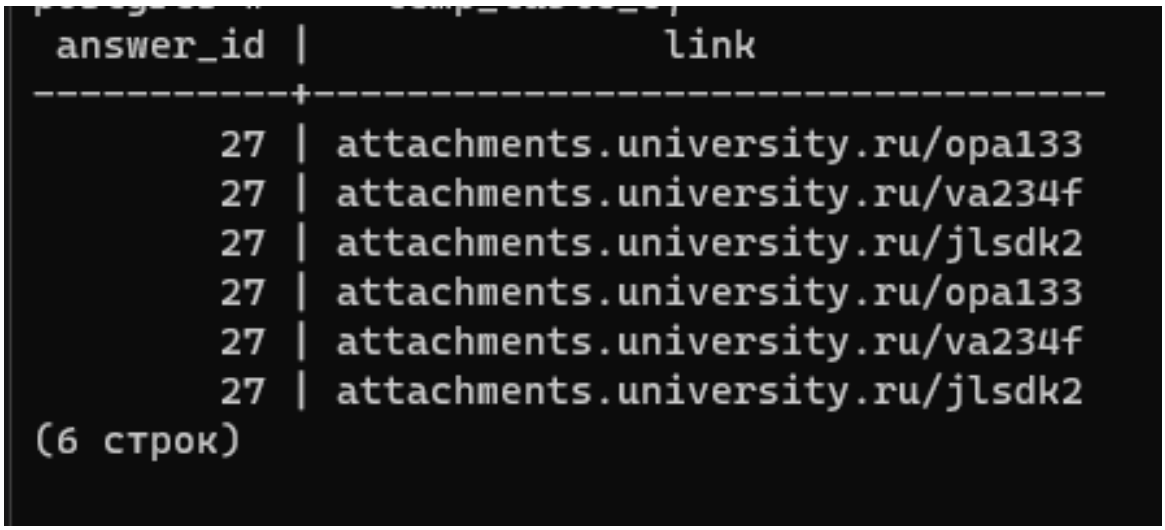
full_name	phone_number	email	web_page
Игорь Владимирович Стольников	+7-931-492-43-97	igorstol@yandex.ru	igorstolpsycho.ru

(1 строка)

Рисунок 6.5 – Результат запроса № 4

Получить приложение, которые прикрепил пользователь к заданному ответу

```sql
SELECT
    a.id AS answer_id,
    att.link AS link
FROM
    answers a
    JOIN attachments att ON a.id = attachments.id
WHERE
    a.id = 27;
```



answer_id	link
27	attachments.university.ru/opa133
27	attachments.university.ru/va234f
27	attachments.university.ru/jlsdk2
27	attachments.university.ru/opa133
27	attachments.university.ru/va234f
27	attachments.university.ru/jlsdk2

(6 строк)

Рисунок 6.6 – Результат запроса № 5

6.6 Оценка размеров базы данных и каждого из файлов

Для оценки размеров базы данных будем основываться на следующих принципах:

- размер одной записи i-таблицы определяется суммой размеров составляющих ее типов данных;
- количество записей в таблице соответствует средним значениям мощностных характериских, описанных ранее.

Рассчетные значения представлены в виде таблицы 6.1.

Таблица 6.1 – Рассчетные значения размеров БД

Таблица	**Объем записи**	**Кол-во записей**	**Объем таблицы**
users	148 байт	6400	947200 байт
roles	12 байт	5	60 байт
users_roles	24 байта	6400	153600 байт
programs	66 байт	80	5280 байт
students	308 байт	6000	184800 байт
teachers	415 байт	300	124500 байт
psychologists	750 байт	20	15000 байт
documents	301 байт	19200	5779200 байт
groups	66 байт	300	19800 байт
groups_students	24 байта	6000	144000 байт
tests	395 байт	1000	395000 байт
questions	500 байт	30000	15000000 байт
tests_questions	24 байта	60000	1440000 байт
tresults	200 байт	60000	12000000 байт
answers	300 байт	1800000	540000000 байт
attachments	271 байт	18000	4878000 байт
subjects	263 байта	200	62600 байт
subjects_groups	36 байт	1200	43200 байт
tests_subjects	24 байта	2500	60000 байт
Итого			581252240 байт

Таким образом, теоритическая оценка базы данных составляет порядка 0.58 гигабайт.

6.7 Архитектура системы

Клиент-серверная архитектура представляет собой структуру взаимодействия между программными компонентами, где одна сторона (клиент) запрашивает услуги или ресурсы, а другая сторона (сервер) предоставляет эти услуги или ресурсы. Вот основные принципы и характеристики этой архитектуры:

- **Распределение обязанностей**.

  Клиент: Отвечает за инициацию запросов и обработку ответов. Взаимодействует с пользователем.

  Сервер: Предоставляет услуги, обрабатывает запросы от клиента, выполняет бизнес-логику и возвращает результаты клиенту.

- **Коммуникация по сети**
  Клиент и сервер обмениваются данными через сетевое соединение. Обычно используется протокол вроде HTTP, TCP/IP или других сетевых протоколов.

- **Масштабируемость**
  Клиенты и серверы могут быть развернуты на разных устройствах или серверах, что обеспечивает гибкость и возможность масштабирования системы.

- **Безопасность**
  Для обеспечения безопасности передачи данных между клиентом и сервером могут использоваться различные механизмы, такие как шифрование и аутентификация.

- **Ответственность**
  Клиент и сервер могут быть реализованы на разных языках программирования и выполняться на разных платформах, что позволяет разделять ответственность между ними.

- **Легкость обновлений**
  При необходимости обновления клиента или сервера, их можно обновить независимо друг от друга, что облегчает поддержку и развитие системы.

- **Распределенная модель**
  Клиент и сервер могут работать на разных физических или логических устройствах, обмениваясь данными по сети. Это позволяет создавать масштабируемые и гибкие системы.

Следует отметить, что приведенные ранее системные требования в полной мере удовлетворяются клиент-серверной архитектурой, поэтому предлагается придерживаться этого подхода к разработке систем в дальнейшем. На рисунке 6.7 приведена концептуальная схема архитектуры системы.

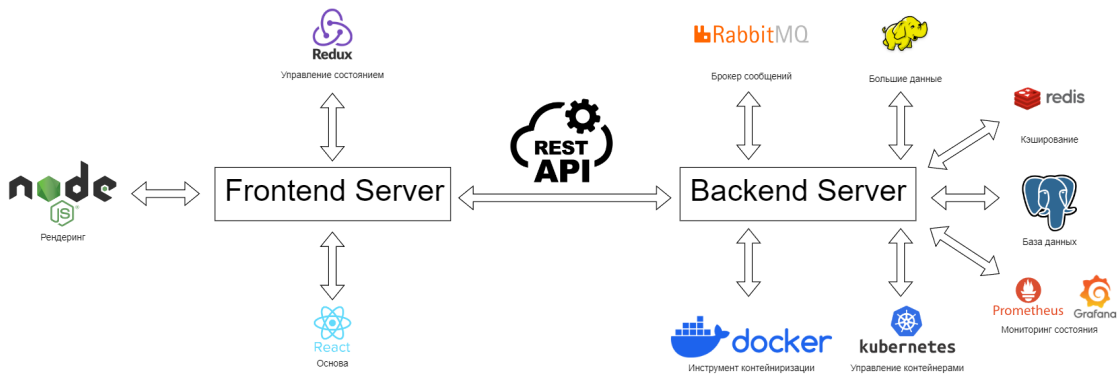


Рисунок 6.7 – Концептуальная схема архитектуры системы

6.8 Отчеты системы

Классификация темпераментов студентов

Знание темперамента человека важно, поскольку оно предоставляет ключевую информацию о его индивидуальных чертах характера, поведенческих предпочтениях и реакциях на стресс. Эта информация полезна как для самопонимания, так и для более эффективного взаимодействия с другими. В профессиональном контексте она способствует лучшему выбору карьеры, эффективной командной работе и разрешению конфликтов. Также знание темперамента может оказать положительное воздействие на личные отношения, улучшив общение и снизив уровень конфликтов.

Диаграмма классов разделения студентов по темпераментам представляет собой красочное изображение студенческого общества в двухмерном признаковом пространстве (см. рисунок 6.8). Каждая точка на диаграмме представляет отдельного студента, а их цвета — их темперамент. Характерные границы разделения подчеркивают, как классификатор выделяет группы студентов с различными темпераментами.

Области, где цвета смешиваются или границы нечетки, могут указывать на переходные зоны между темпераментами. Такие детали предоставляют уникальный взгляд на структуру студенческого общества и могут быть полезными для понимания распределения темпераментов внутри группы.

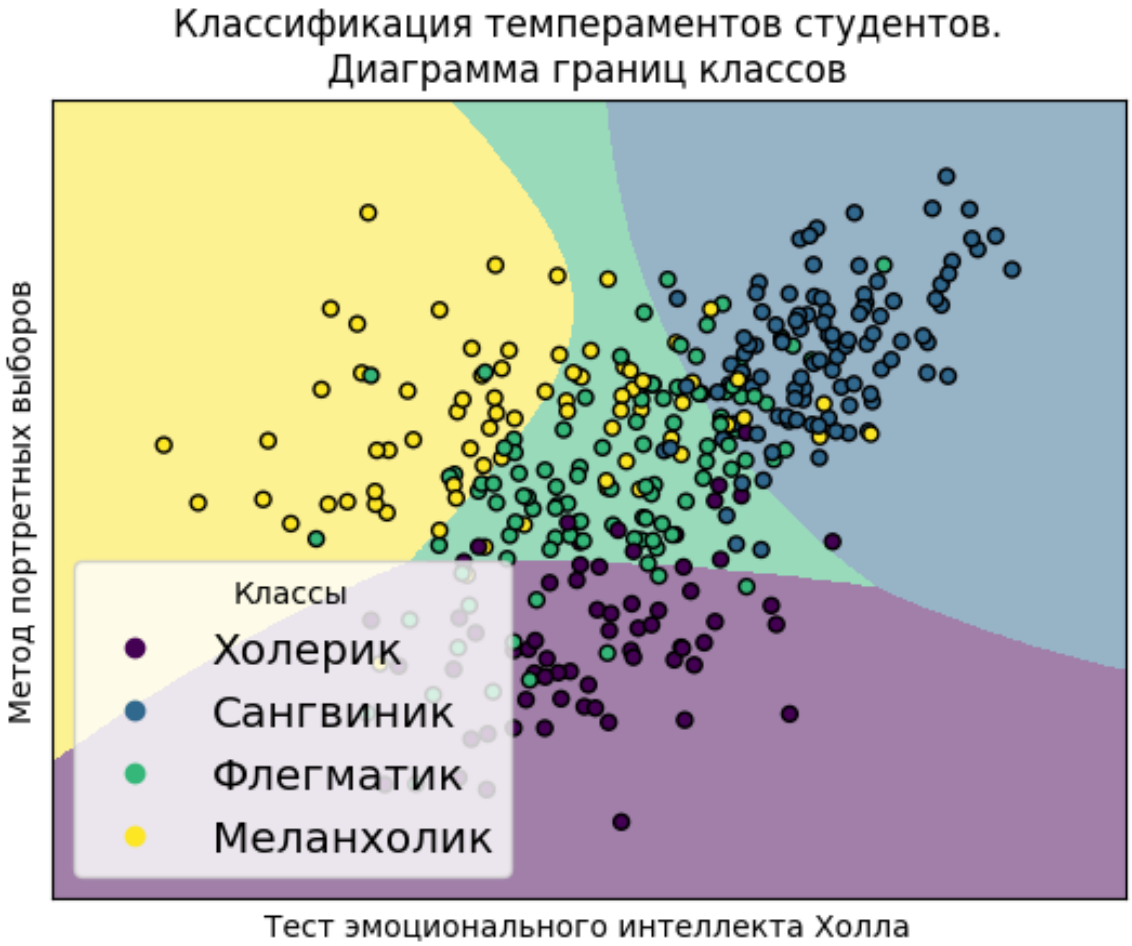


Рисунок 6.8 – Диаграмма границ классов

Мониторинг состояния системы

Мониторинг состояния системы играет критическую роль в обеспечении ее стабильной и эффективной работы. Раннее обнаружение проблем позволяет предотвращать сбои и минимизировать последствия. Отслеживание производительности системы помогает выявлять и устранять узкие места, обеспечивая более эффективное использование ресурсов.

Эффективный мониторинг позволяет оперативно реагировать на инциденты, уменьшая время простоя системы. Предсказание тенденций использования ресурсов помогает планировать масштабирование и предотвращать исчерпание ресурсов. Важен также аспект безопасности — мониторинг обнаруживает аномалии и потенциальные угрозы, способствуя обеспечению безопасности системы.

В целом, мониторинг состояния системы не только предотвращает проблемы, но и повышает ее надежность, производительность и безопасность,

что является фундаментальным элементом успешного и устойчивого функционирования любой системы.

Промышленные требования к мониторингу систем сделали инструменты, такие как Prometheus и Grafana, неотъемлемой частью современных практик DevOps. Prometheus является мощным инструментом сбора метрик, основанным на модели pull-мониторинга. Он активно собирает информацию о состоянии системы, предоставляя обширные данные о производительности, использовании ресурсов и других ключевых параметрах. Гибкость Prometheus заключается в том, что он легко настраивается, а экспортеры позволяют интегрировать его с различными приложениями.

Grafana, в свою очередь, обеспечивает визуализацию этих данных, предоставляя интуитивно понятные и гибкие дашборды. Он не только позволяет наглядно отслеживать работу системы, но и предоставляет средства алертинга для оперативного реагирования на инциденты. Преимущества данного подхода включают гибкость, расширяемость и использование открытых стандартов, что делает Prometheus и Grafana востребованными инструментами в области мониторинга и управления системами. С их помощью команды DevOps могут эффективно выявлять и решать проблемы, оптимизировать производительность системы и обеспечивать стабильную и безопасную работу в различных сценариях использования.

Так на рисунке 6.9 приведен пример экрана мониторинга состояния БД при помощи средств Prometheus + Grafana.

Рисунок 6.9 – Пример экрана мониторинга состояния БД

Мониторинг успеваемости

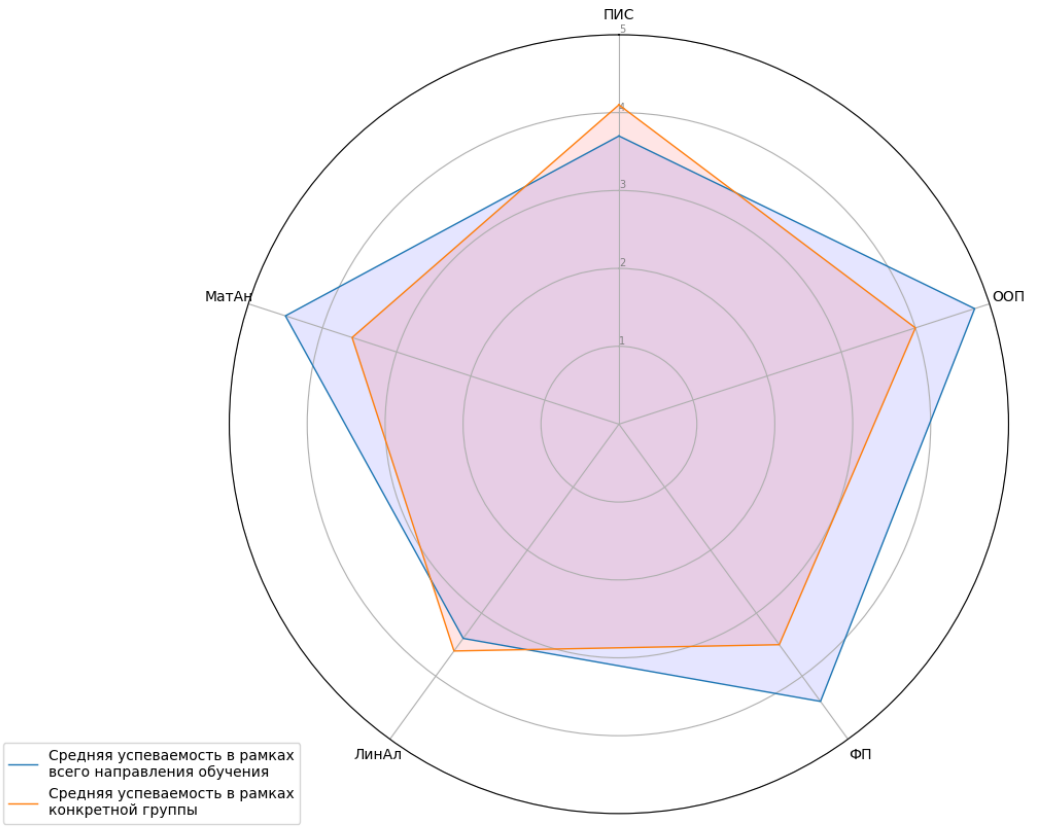


Рисунок 6.10 – Оценка средней успеваемости группы

Мониторинг успеваемости студентов является ключевым элементом образовательного процесса, обеспечивая множество выгод. Во-первых, это предоставляет преподавателям и учебным заведениям информацию о степени понимания учебного материала студентами, что позволяет им адаптировать методику обучения для лучшего усвоения информации. Во-вторых, мониторинг успеваемости является инструментом для выявления возможных проблем и предоставляет студентам обратную связь, способствуя их академическому росту и развитию. Такой подход способствует эффективной адаптации учебных программ, повышает качество образования и способствует успеху студентов в их учебном пути.

На рисунке 6.10 приведен пример оценки средней успеваемости группы по сравнению со студентами всего направления обучения, благодаря чему можно выявить проблемные зоны и устранить их в дальнейшем.