**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

****

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

|  |  |
| --- | --- |
| Институт | компьютерных наук |
| Кафедра | автоматизированных систем управления |

КУРСОВОЙ ПРОЕКТ

по дисциплине «Разработка информационной системы»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

«Разработка информационной системы»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ПИ-22-1 | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | Насонов Н.С. |
| Руководитель | |  |  | Алексеев В.А. |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата |

Липецк, 2025 г.

**Задание для курсового проекта**

Разработать веб-приложение для автоматизации процесса начисления и распределения стипендий среди студентов образовательного учреждения, а также автоматизации связанных для данного процесса взаимодействий пользователя.

# Оглавление

[Оглавление 3](#_Toc199343111)

[Введение 6](#_Toc199343112)

[1 Техническое задание 7](#_Toc199343113)

[1.1 Характеристика предметной области 7](#_Toc199343114)

[1.2 Цели и задачи системы 7](#_Toc199343115)

[1.3 Автоматизируемые бизнес-процессы 8](#_Toc199343116)

[1.4 Характеристика пользователей 9](#_Toc199343117)

[1.5 Модель вариантов использования 10](#_Toc199343118)

[1.6 Модели бизнес-процессов в нотации BPMN 11](#_Toc199343119)

[1.7 Требования к информационному обеспечению 12](#_Toc199343120)

[1.8 Требования к программному обеспечению и пользовательскому интерфейсу 13](#_Toc199343121)

[2 Технический проект 15](#_Toc199343122)

[2.1 Модели локальных представлений 15](#_Toc199343123)

[2.1.1 Локальные ER-диаграммы 15](#_Toc199343124)

[2.1.2 Основные запросы 18](#_Toc199343125)

[2.1.3 Общая ER-диаграмма 19](#_Toc199343126)

[2.2 Концептуальная модель данных 20](#_Toc199343127)

[2.2.1 ER-диаграмма 20](#_Toc199343128)

[2.2.2 Спецификация сущностей 21](#_Toc199343129)

[2.2.3 Спецификация связей 24](#_Toc199343130)

[2.3 Логическая модель данных 25](#_Toc199343131)

[2.3.1 Диаграмма логической модели 25](#_Toc199343132)

[2.3.2 Спецификация логической модели 26](#_Toc199343133)

[2.4 Физическая модель данных 30](#_Toc199343134)

[2.4.1 Обоснование выбора СУБД 30](#_Toc199343135)

[2.4.2 Диаграмма физической модели 31](#_Toc199343136)

[2.4.3 Спецификация таблиц 32](#_Toc199343137)

[2.4.4 Проектирование вторичных индексов 36](#_Toc199343138)

[2.5 Проектирование запросов выборки 37](#_Toc199343139)

[2.5.1 Запрос №1 37](#_Toc199343140)

[2.5.2 Запрос №2 38](#_Toc199343141)

[2.5.3 Запрос №3 39](#_Toc199343142)

[2.5.4 Запрос №4 40](#_Toc199343143)

[2.5.5 Запрос №5 41](#_Toc199343144)

[2.6 Пользовательские представления 42](#_Toc199343145)

[2.6.1 Представление №1 42](#_Toc199343146)

[2.6.2 Представление №2 44](#_Toc199343147)

[2.7 Архитектура информационной системы 46](#_Toc199343148)

[2.7.1 Диаграмма компонентов 46](#_Toc199343149)

[2.7.2 Спецификация компонентов 46](#_Toc199343150)

[2.7.3 Распределение бизнес-логики между компонентами 47](#_Toc199343151)

[2.7.4 Интерфейсы взаимодействия компонентов 47](#_Toc199343152)

[2.8 Хранимые процедуры и триггеры 48](#_Toc199343153)

[2.8.1 Хранимая процедура №1 48](#_Toc199343154)

[2.8.2 Триггер №1 52](#_Toc199343155)

[2.8.3 Триггер №2 54](#_Toc199343156)

[3 Рабочий проект 56](#_Toc199343157)

[3.1 SQL-скрипт создания структуры БД 56](#_Toc199343158)

[3.2 SQL-скрипт триггеров и хранимых процедур 56](#_Toc199343159)

[3.3 Текст программы 56](#_Toc199343160)

[3.4 Руководство пользователя 56](#_Toc199343161)

[Заключение 57](#_Toc199343162)

# Введение

Веб-приложение для автоматизации процесса начисления и распределения стипендий между студентами образовательного учреждения, направлено на ускорение взаимодействия всех участников образовательного учреждения между собой.

Распределение стипендиального бюджета представляет собой трудоемкий процесс сбора, анализа и обработки многих факторов для назначения стипендий. Приложения позволяет автоматизировать сбор информации от различных категорий пользователей. Эта информация может в себя включать достижения студентов, оценки, выставленные студентам, стипендиальный бюджет на следующий семестр, а также личная информация всех участников системы.

Таким образом реализация веб-приложения для автоматизации процесса начисления и распределения стипендий способствует упрощению расчёта стипендий и повышения скорости и качества взаимодействия между пользователями.

1. Техническое задание

## Характеристика предметной области

Предметная области представляет собой систему расчета стипендий «Стипендиатус» занимается распределением и расчётом стипендий в зависимости от бюджета и количества стипендиатов.

В данной предметной области необходимо учитывать особенности назначения стипендий из стипендиального бюджета, а также правильное распределение между всеми участниками системы. Для этого необходимо учесть каким участникам системы могут быть назначены выплаты, а также как распределять остатки бюджета между различными участниками в зависимости от их рейтинга.

Ключевые аспекты:

* Предоставление возможности внесения достижений студенту, просмотра оценок и назначенных выплат
* Эффективное распределение бюджета
* Поддержка обратной связи между всеми участниками образовательного учреждения

## Цели и задачи системы

Целью данной системы является улучшение и оптимизация назначения стипендий и распределения бюджета, предоставление удобного пользовательского интерфейса для качественной работы с системой, а также обеспечения эффективного управления процессом взаимодействия между участники образовательного учреждения.

Автоматизированная информационная система предназначена для автоматизации расчёта и распределения стипендии между студентами, а также предоставления отчетов и информации о стипендиях бухгалтеру, заместителям декана, студентам.

## Автоматизируемые бизнес-процессы

1. Внесение данных о студентах — деканат вводит или обновляет данные о студентах, включая их личную информацию и данные об успеваемости. Студент может отправить запрос на изменение данных при неточностях
2. Расчёт стипендии — на основе успеваемости студентов и их достижений производится автоматический расчет стипендий (базовых и повышенных) с учетом выделенного бюджета и общего количества стипендий (повышенных).
3. Предоставление отчетов — система формирует отчеты для бухгалтерии (сводные данные о бюджете и распределении средств), деканата (успеваемость студентов) и студентов (информация о назначенных стипендиях и их изменениях).
4. Внесение изменений в документы – студенты и деканат могут корректировать документы (например, исправления в успеваемости), а система обновляет расчеты.

## Характеристика пользователей

1. Бухгалтерия

Бухгалтерия – просмотр общего бюджета, а также автоматическое распределение стипендий между студентами-бюджетниками

1. Студенты

Студенты – просмотр назначенных стипендий, внесение информации о достижениях, влияющих на нее, а также подача документов на повышенную стипендию в установленной форме. Возможность просмотра информации об успеваемости и достижениях, для запросов об исправлении.

1. Деканат

Деканат – внесение успеваемости студентов и их личной информации, и их личной информации, исправление ошибок в документах.

## Модель вариантов использования

Модель вариантов использования предоставлена на рисунке 1.

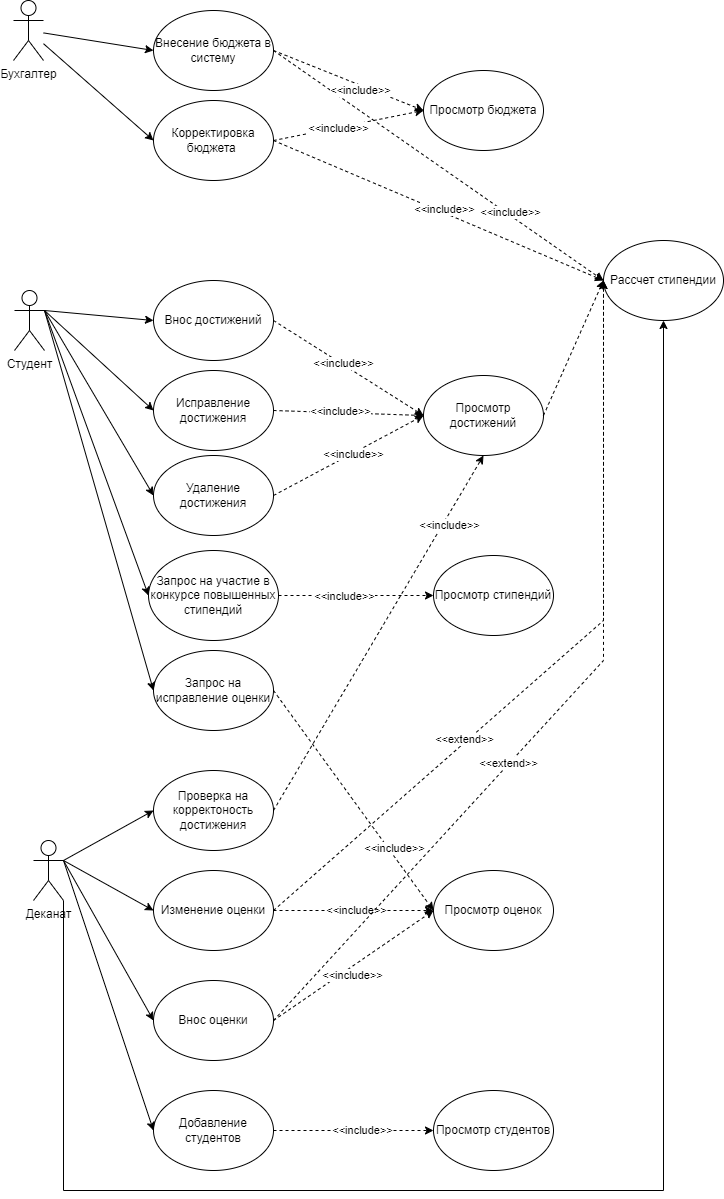


Рисунок 1 – use-case диаграмма

## Модели бизнес-процессов в нотации BPMN

Модель бизнес-процесса утверждения нового плана производства на сутки/определенный период приведена на рисунке 2.

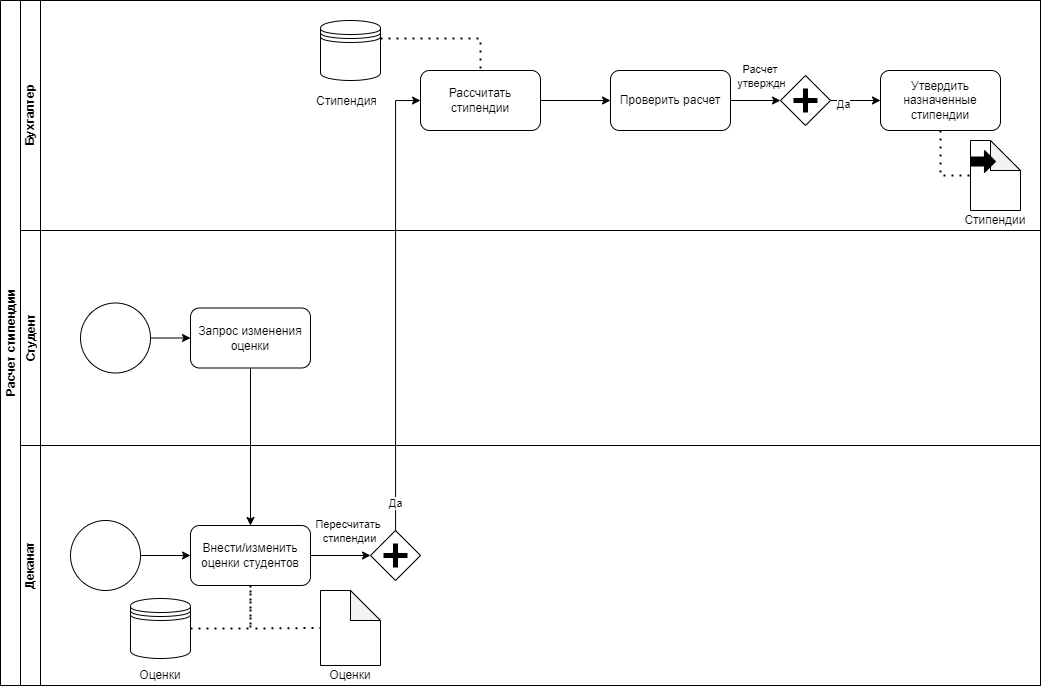


Рисунок 2 – модель BPMN бизнес-процесса

## Требования к информационному обеспечению

1. Концептуальная модель данных должна содержать не менее 5 сущностей.

2. СУБД – PostgreSQL.

3. Физическая схема должна предусматривать реализацию индексов и пользовательского представления.

Информационное обеспечение должно быть достаточным для эффективной работы всех функций системы и обеспечивать совместимость информации между различными компонентами системы. Данные должны храниться на сервере в базе данных для обеспечения быстрой обработки и длительного хранения.

Должны быть предусмотрены меры по контролю, обновлению данных и восстановлению данных в случае сбоев устройств. Доступ к информации должен быть удобным и быстрым.

Выходные документы, должны быть наглядными для удобства восприятия персоналом.

Также важно уделить внимание аутентификации и авторизации пользователей для обеспечения безопасного доступа к данны

## Требования к программному обеспечению и пользовательскому интерфейсу

Обязательные требования к программному обеспечению:

1. Проект должен предусматривать реализацию триггеров и хранимых процедур.

2. Платформа разработки прикладного приложения Visual Studio Code. Язык программирования – Python, Go, Vue, TypeScript. Фреймворки – Flask, GIN. Для доступа к БД используется библиотека-драйвер БД pgx

3. Прикладное приложение должно иметь удобный пользовательский интерфейс, реализующий функции информационной системы, предусмотренные техническим заданием.

4. Прикладное приложение должно скрывать от пользователя технические детали организации данных в БД (искусственные идентификаторы и т.п.).

5. Прикладное приложение должно иметь функционально-ориентированный интерфейс, спроектированный под выполнение задач пользователя (а не под структуру БД).

6. Приложение должно предусматривать генерацию отчетных форм с использованием специализированных библиотек компонентов для разработки отчетов (FastReport или т.п.), с возможностью экспорта отчетов в стандартные форматы (PDF/Excel/Word и т.д.). Приветствуется использование в отчетах графических элементов – диаграмм и т.п.

Прикладное приложение должно содержать следующие элементы интерфейса:

1) Меню для навигации по функциям (левое или верхнее). Для мобильных приложений могут использоваться карточки для навигации по функциям.

2) Табличное представление данных с фильтром и пагинацией. В таблице не должны отображаться искусственные идентификаторы, они должны быть заменены содержательными значениями полей.

3) Редактирование записей в таблице «на месте» – при небольшом числе атрибутов, если горизонтальная прокрутка отсутствует.

4) Редактирование записей в режиме «модального окна» или «боковой панели» (паттерн «контекстный оверлей»). Не должны отображаться искусственные идентификаторы, они должны быть заменены содержательными значениями полей.

5) Выпадающие списки или чек-листы для выбора связанных значений в другой таблице.

6) Календари, переключатели, чекбоксы – там, где это применимо.

7) При формировании отчетов должен быть предусмотрен отбор данных, например, за период времени, по категориям объектов и другим признакам (например, отчет по определенной категории товаров за год и т.п.).

# Технический проект

## Модели локальных представлений

### Локальные ER-диаграммы

Составим локальные ER-диаграммы для АИС «Стипендиатус». По категориям пользователей в нотации Чена. Полученные локальные ER-диаграммы для студента, деканата, бухгалтерии представлены на рисунках 3 – 5.

Изображение выглядит как диаграмма, зарисовка, рисунок, шаблон

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3 – Локальное представление со стороны бухгалтерии

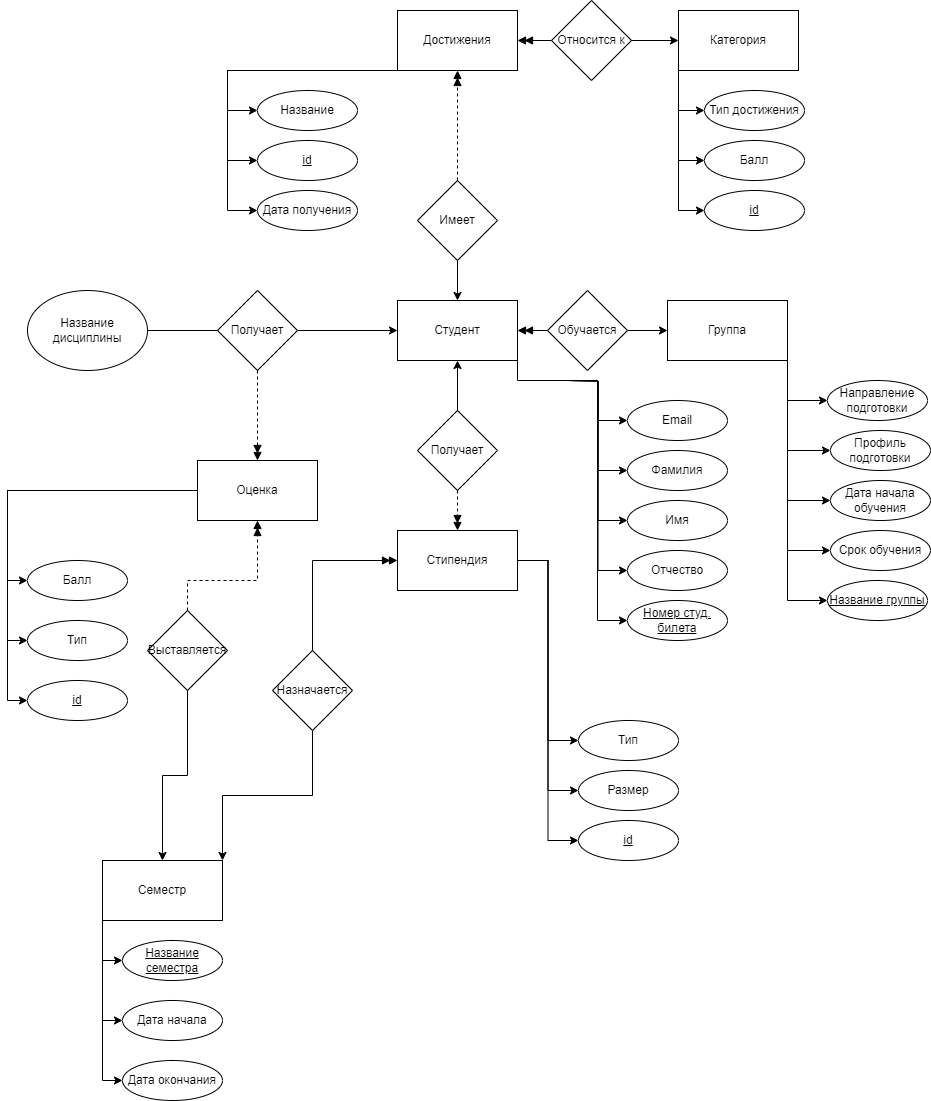


Рисунок 4 – Локальное представление со стороны студента

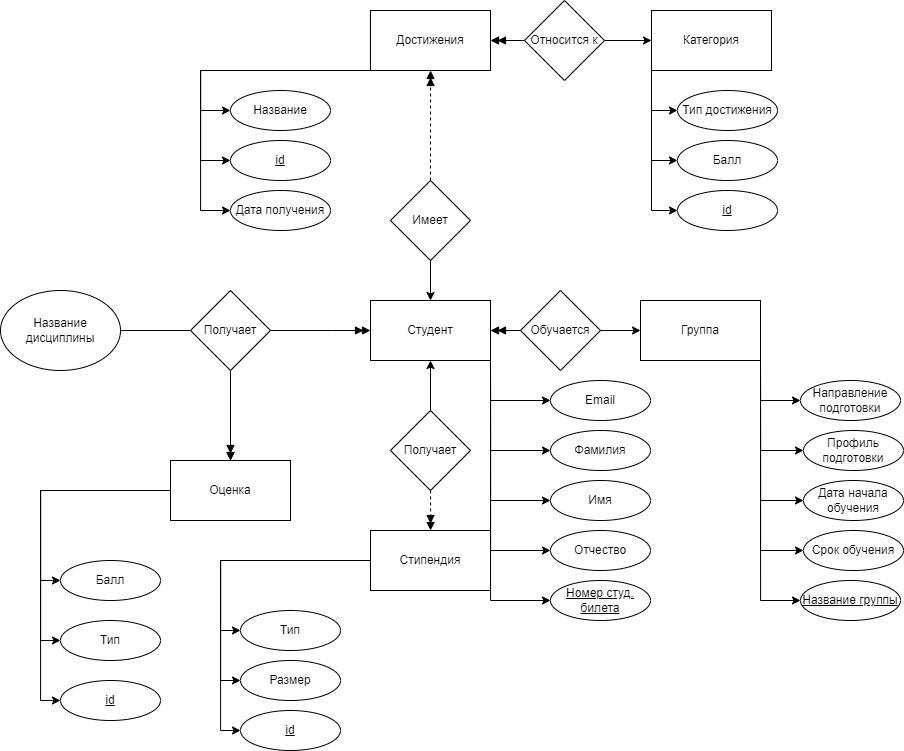


Рисунок 5 – Локальное представление со стороны студента

### Основные запросы

1. Основные запросы для студентов:
   1. Получение назначенных стипендий
   2. Получение выставленных оценок
   3. Добавление достижений
   4. Авторизация аккаунта
2. Основные запросы для деканата:
   1. Просмотр, редактирование и создание студентов
   2. Просмотр, редактирование и выставление оценок
   3. Просмотр, редактирование и создание достижений
3. Основные запросы для бухгалтеров:
   1. Просмотр и редактирование бюджетов
   2. Просмотр и редактирование назначенных стипендий

### Общая ER-диаграмма

На рисунке 6 представлена общая ER-диаграмма в нотации Чена.

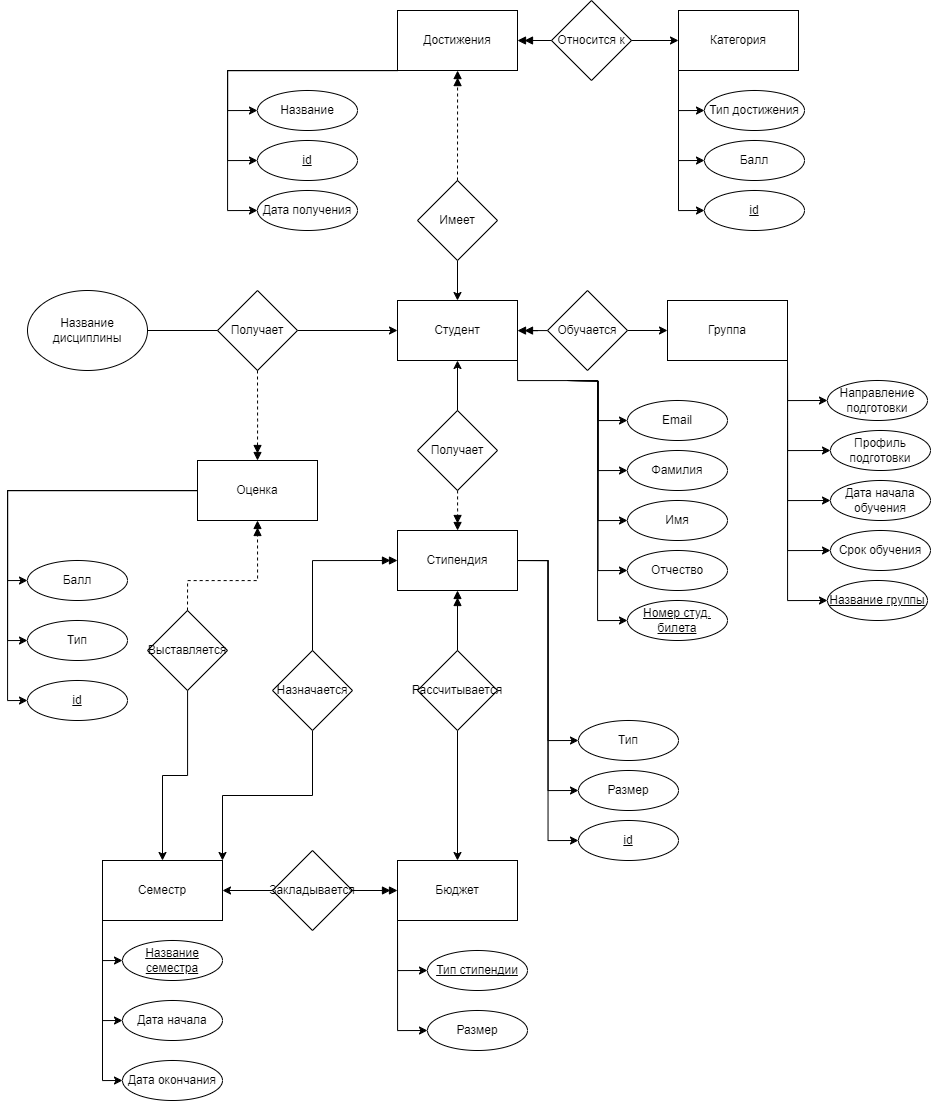


Рисунок 6 – Общая ER-диаграмма

## Концептуальная модель данных

### ER-диаграмма

На рисунке 7 представлена концептуальная модель данных в нотации Craw’s Foot.



Рисунок 7 – Концептуальная модель данных

### Спецификация сущностей

По концептуальной модели, представленной на рисунке 7, в пункте 2.2.1, составлены спецификации всех сущностей. Спецификации сущностей отображены в таблицe 1.

Таблица 1 – спецификация сущностей

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Наименование сущности** | **Код сущности** | **Наименование атрибута** | **Код атрибута** | **Тип данных** | **Обязательность** | **Первичный ключ** |
| Достижение | Achievement | id | id\_achivment | Serial | X | X |
| Название | name\_achivement | Variable characters (200) | X |  |
| Дата получения | date\_achivment | Date | X |  |
| Бюджет | Budget | Тип стипендии | type\_scholarship\_budget | Variable characters (100) | X | X |
| Размер | size\_budget | Money | X |  |
| Категория | Category | id | id\_category | Serial | X | X |
| Тип достижения | achivments\_type\_category | Variable characters (50) | X |  |
| Балл | score\_category | Short integer | X |  |
| Группа | Group | Название группы | name\_group | Variable characters (10) | X | X |
| Направление подготовки | studies\_direction\_group | Variable characters (8) | X |  |
| Профиль подготовки | studies\_profile\_group | Variable characters (50) | X |  |
| Дата начала обучения | start\_date\_group | Date | X |  |
| Срок обучения | studies\_period-group | Short integer | X |  |

Продолжение таблицы 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Наименование сущности** | **Код сущности** | **Наименование атрибута** | **Код атрибута** | **Тип данных** | **Обязательность** | **Первичный ключ** |
| Оценка | mark | id | id\_mark | Serial | X | X |
| Название дисциплины | lesson\_name\_mark | Variable characters (100) | X |  |
| Балл | score\_mark | Short integer | X |  |
| Тип | type\_mark | Variable characters (40) | X |  |
| Стипендия | Scholarship | id | id\_scholarship | Serial | X | X |
| Размер | size\_scholarshp | Money | X |  |
| Тип | type\_scholarship | Variable characters (100) | X |  |
| Семестр | Semester | Название семестра | name\_semester | Variable characters (5) | X | X |
| Дата начала | date\_start\_semester | Date | X |  |
| Дата окончания | date\_end\_semester | Date | X |  |
| Студент | Student | Номер студ. билета | id\_num\_student | Integer | X | X |
| Email | email\_student | Variable characters (40) |  |  |
| Фамилия | second\_name\_student | Variable characters (40) | X |  |
| Имя | first\_name\_student | Variable characters (40) | X |  |
| Отчество | surname\_student | Variable characters (40) |  |  |

Окончание таблицы 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Наименование сущности** | **Код сущности** | **Наименование атрибута** | **Код атрибута** | **Тип данных** | **Обязательность** | **Первичный ключ** |
| Пользователь | User | user\_id | user\_id | Serial | X | X |
| Логин | user\_login | Text | X |  |
| user\_email | user\_email | Text | X |  |
| Роль | user\_role | Text | X |  |
| Пароль | user\_password | Text | X |  |
| user\_student\_id | user\_student\_id | Number |  |  |

### Спецификация связей

По концептуальной модели, представленной на рисунке 7, в пункте 2.2.1, составлена спецификация связей. Спецификация связей отображена в таблице 2.

Таблица 2 – спецификация связей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Сущность 1** | **Сущность 2** | **Наименование связи** | **Код связи** | **Вид связи** |
|
| Semester | Scholarship | Назначается | assign | 1,n |
| Budget | Scholarship | Рассчитывает | Calculate | 1,n |
| Student | mark | Получает | Get | 0,n |
| Semester | mark | Выставляется | give | 0,n |
| Student | Achievement | Имеет | Have | 0,n |
| Semester | Budget | Закладывается | lay | 1,n |
| Student | Scholarship | Получает | Receive | 0,n |
| Category | Achievement | Относится | Refer | 1,n |
| Group | Student | Обучается | Study | 1,n |

## Логическая модель данных

### Диаграмма логической модели

На основе концептуальной модели, изображенной на рисунке 7, находящемся в пункте 2.2.1, построена логическая модель данных в нотации Craw`s Foot. Логическая модель данных изображена на рисунке 8.



Рисунок 8 – Логическая модель в нотации Craw`s Foot с обозначением name

### Спецификация логической модели

По логической модели, представленной на рисунке 8, в пункте 2.3.1, составлена спецификация логической модели. Спецификация логической модели отображена в таблице 3.

Таблица 3 – спецификация логической модели

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Наименование таблицы** | **Код таблицы** | **Наименование атрибута** | **Код атрибута** | **Тип данных** | **Первичный ключ** | **Вторичный ключ** | **Обязательность** |
| Достижение | Achievement | id | id\_achivment | Serial | **+** | **-** | X |
| Номер студ. Билета | id\_num\_student | Integer | **-** | **+** | X |
| id2 | id\_category | Integer | **-** | **+** | X |
| Название | name\_achivement | Variable characters (200) | **-** | **-** | X |
| Дата получения | date\_achivment | Date | **-** | **-** | X |
| Бюджет | Budget | Тип стипендии | type\_scholarship\_budget | Variable characters (100) | **+** | **-** | X |
| Название семестра | name\_semester | Variable characters (5) | **-** | **+** | X |
| Размер | size\_budget | Money | **-** | **-** | X |
| Категория | Category | id | id\_category | Serial | **+** | **-** | X |
| Тип достижения | achivments\_type\_category | Variable characters (50) | **-** | **-** | X |
| Балл | score\_category | Short integer | **-** | **-** | X |

Продолжение таблицы 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Наименование таблицы** | **Код таблицы** | **Наименование атрибута** | **Код атрибута** | **Тип данных** | **Первичный ключ** | **Вторичный ключ** | **Обязательность** |
| Группа | Group | Название группы | name\_group | Variable characters (10) | **+** | **-** | X |
| Направление подготовки | studies\_direction\_group | Variable characters (8) | **-** | **-** | X |
| Профиль подготовки | studies\_profile\_group | Variable characters (50) | **-** | **-** | X |
| Дата начала обучения | start\_date\_group | Date | **-** | **-** | X |
| Срок обучения | studies\_period-group | Short integer | **-** | **-** | X |
| Оценка | mark | id | id\_mark | Serial | **+** | **-** | X |
| Номер студ. билета | id\_num\_student | Integer | **-** | **+** | X |
| Название семестра | name\_semester | Variable characters (5) | **-** | **+** | X |
| Название дисциплины | lesson\_name\_mark | Variable characters (100) | **-** | **-** | X |
| Балл | score\_mark | Short integer | **-** | **-** | X |
| Тип | type\_mark | Variable characters (40) | **-** | **-** | X |

Продолжение таблицы 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Наименование таблицы** | **Код таблицы** | **Наименование атрибута** | **Код атрибута** | **Тип данных** | **Первичный ключ** | **Вторичный ключ** | **Обязательность** |
| Стипендия | Scholarship | id | id\_scholarship | Serial | **+** | **-** | X |
| Номер студ. билета | id\_num\_student | Integer | **-** | **+** | X |
| Тип стипендии | type\_scholarship\_budget | Variable characters (100) | **-** | **+** | X |
| Название семестра | name\_semester | Variable characters (5) | **-** | **-** | X |
| Размер | size\_scholarshp | Money | **-** | **-** | X |
| Тип | type\_scholarship | Variable characters (100) | **-** | **-** | X |
| Семестр | Semester | Название семестра | name\_semester | Variable characters (5) | **+** | **-** | X |
| Дата начала | date\_start\_semester | Date | **-** | **-** | X |
| Дата окончания | date\_end\_semester | Date | **-** | **-** | X |

Окончание таблицы 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Наименование таблицы** | **Код таблицы** | **Наименование атрибута** | **Код атрибута** | **Тип данных** | **Первичный ключ** | **Вторичный ключ** | **Обязательность** |
| Студент | Student | Номер студ. билета | id\_num\_student | Integer | **+** | **-** | X |
| Название группы | name\_group | Variable characters (10) | **-** | **+** | X |
| Email | email\_student | Variable characters (40) | **-** | **-** |  |
| Фамилия | second\_name\_student | Variable characters (40) | **-** | **-** | X |
| Имя | first\_name\_student | Variable characters (40) | **-** | **-** | X |
| Отчество | surname\_student | Variable characters (40) | **-** | **-** |  |
| Пользователь | User | user\_id | user\_id | Serial | + | - | X |
| Логин | user\_login | Text | **-** | **-** | X |
| user\_email | user\_email | Text | **-** | **-** | X |
| Роль | user\_role | Text | **-** | **-** | X |
| Пароль | user\_password | Text | **-** | **-** | X |
| user\_student\_id | user\_student\_id | Number | **-** | **-** |  |

## Физическая модель данных

### Обоснование выбора СУБД

Была выбрана СУБД PostgreSQL так как она имеет ряд особенностей.

1. Имеет полную поддержку ключей, объединений, представлений, триггеров;
2. Популярная и надежная;
3. Бесплатная;
4. Легко масштабируется;
5. Поддержка БД практически неограниченного размера.

### Диаграмма физической модели

На основе логической модели, изображенной на рисунке 8, находящемся в пункте 2.3.1, построена физическая модель данных в нотации Craw`s Foot. Физическая модель данных изображена на рисунке 9.



Рисунок 9 – Физическая модель данных для СУБД PostgreSQL

### Спецификация таблиц

По физической модели, представленной на рисунке 8, в пункте 2.4.1, составлена спецификация таблиц. Спецификация таблиц отображена в таблице 4.

Таблица 4 – спецификация таблиц

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Наименование таблицы** | **Код таблицы** | **Наименование атрибута** | **Код атрибута** | **Тип данных** | **Первичный ключ** | **Вторичный ключ** | **Обязательность** |
| Достижение | Achievement | id | id\_achivment | SERIAL | **+** | **-** | X |
| Номер студ. билета | id\_num\_student | INT4 | **-** | **+** | X |
| id2 | id\_category | INT4 | **-** | **+** | X |
| Название | name\_achivement | VARCHAR(200) | **-** | **-** | X |
| Дата получения | date\_achivment | DATE | **-** | **-** | X |
| Бюджет | Budget | Тип стипендии | type\_scholarship\_budget | VARCHAR(100) | **+** | **-** | X |
| Название семестра | name\_semester | VARCHAR(5) | **-** | **+** | X |
| Размер | size\_budget | MONEY | **-** | **-** | X |
| Категория | Category | id | id\_category | SERIAL | **+** | **-** | X |
| Тип достижения | achivments\_type\_category | VARCHAR(50) | **-** | **-** | X |
| Балл | score\_category | INT2 | **-** | **-** | X |

Продолжение таблицы 4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Наименование таблицы** | **Код таблицы** | **Наименование атрибута** | **Код атрибута** | **Тип данных** | **Первичный ключ** | **Вторичный ключ** | **Обязательность** |
| Группа | Group | Название группы | name\_group | VARCHAR(10) | **+** | **-** | X |
| Направление подготовки | studies\_direction\_group | VARCHAR(8) | **-** | **-** | X |
| Профиль подготовки | studies\_profile\_group | VARCHAR(50) | **-** | **-** | X |
| Дата начала обучения | start\_date\_group | DATE | **-** | **-** | X |
| Срок обучения | studies\_period-group | INT2 | **-** | **-** | X |
| Оценка | mark | id | id\_mark | SERIAL | **+** | **-** | X |
| Номер студ. билета | id\_num\_student | INT4 | **-** | **+** | X |
| Название семестра | name\_semester | VARCHAR(5) | **-** | **+** | X |
| Название дисциплины | lesson\_name\_mark | VARCHAR(100) | **-** | **-** | X |
| Балл | score\_mark | INT2 | **-** | **-** | X |
| Тип | type\_mark | VARCHAR(40) | **-** | **-** | X |

Продолжение таблицы 4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Наименование таблицы** | **Код таблицы** | **Наименование атрибута** | **Код атрибута** | **Тип данных** | **Первичный ключ** | **Вторичный ключ** | **Обязательность** |
| Стипендия | Scholarship | id | id\_scholarship | SERIAL | **+** | **-** | X |
| Номер студ. билета | id\_num\_student | INT4 | **-** | **+** | X |
| Тип стипендии | type\_scholarship\_budget | VARCHAR(100) | **-** | **+** | X |
| Название семестра | name\_semester | VARCHAR(5) | **-** | **-** | X |
| Размер | size\_scholarshp | MONEY | **-** | **-** | X |
| Тип | type\_scholarship | VARCHAR(100) | **-** | **-** | X |
| Семестр | Semester | Название семестра | name\_semester | VARCHAR(5) | **+** | **-** | X |
| Дата начала | date\_start\_semester | DATE | **-** | **-** | X |
| Дата окончания | date\_end\_semester | DATE | **-** | **-** | X |
| Студент | Student | Номер студ. билета | id\_num\_student | INT4 | **+** | **-** | X |
| Название группы | name\_group | VARCHAR(10) | **-** | **+** | X |
| Email | email\_student | VARCHAR(40) | **-** | **-** | X |
| Фамилия | second\_name\_student | VARCHAR(40) | **-** | **-** | X |
| Имя | first\_name\_student | VARCHAR(40) | **-** | **-** | X |
| Отчество | surname\_student | VARCHAR(40) | **-** | **-** |  |

Окончание таблицы 4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Наименование таблицы** | **Код таблицы** | **Наименование атрибута** | **Код атрибута** | **Тип данных** | **Первичный ключ** | **Вторичный ключ** | **Обязательность** |
| Пользователь | User | user\_id | user\_id | Serial | + | - | X |
| Логин | user\_login | Text | **-** | **-** | X |
| user\_email | user\_email | Text | **-** | **-** | X |
| Роль | user\_role | Text | **-** | **-** | X |
| Пароль | user\_password | Text | **-** | **-** | X |
| user\_student\_id | user\_student\_id | Number | **-** | **-** |  |

### Проектирование вторичных индексов

Вторичный индекс для таблицы Mark по полю score:

Обоснование: Индекс по полю score будет полезен для оптимизации расчётов стипендии, где требуется средний балл по всем дисциплинам студента.

create index average\_score\_IDX on mark (

name\_semester,

lesson\_name\_mark,

type\_mark,

score\_mark

);

## Проектирование запросов выборки

### Запрос №1

Получение среднего балла студента с id\_num\_student = 1220060404 и сравнение его с фиксированным значением. Результат выполнения представлен на рисунке 10.

Реляционная алгебра

Student1220 ← σ\_{id\_num\_student = 1220060404}(student)

AvgMark1220 ← γ\_{id\_num\_student; AVG(score\_mark) → avg\_mark}(σ\_{id\_num\_student = 1220060404}(mark))

Result ← Student1220 ⨝ AvgMark1220

π\_{first\_name\_student, second\_name\_student, avg\_mark}(Result)

SQL скрипт

SELECT

s.first\_name\_student,

s.second\_name\_student,

(SELECT AVG(m.score\_mark)

FROM mark m

WHERE m.id\_num\_student = 1220060404) AS avg\_mark

FROM student s

WHERE id\_num\_student = 1220060404;

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 10 – Результат выполнения запроса

### Запрос №2

Количество студентов в каждой группе. Результат выполнения представлен на рисунке 11.

Реляционная алгебра

γ\_{name\_group; COUNT(\*) → students\_count}(student)

SQL скрипты

SELECT name\_group, COUNT(\*) AS students\_count

FROM student

GROUP BY name\_group;

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 11– Результат выполнения запроса

Запрос №3

Выбрать студентов, у которых есть хотя бы одна оценка выше 85. Результат выполнения представлен на рисунке 12.

Реляционная алгебра

Result ← student ⋉\_{student.id\_num\_student = mark.id\_num\_student ∧ mark.score\_mark > 85} mark

SQL скрипт

SELECT \*

FROM student s

WHERE EXISTS (

SELECT 1

FROM mark m

WHERE m.id\_num\_student = s.id\_num\_student

AND m.score\_mark > 85

);

Изображение выглядит как текст, Шрифт, число, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 12 – Результат выполнения запроса

Запрос №4

Получение студентов за определённый семестр со стипендией в пределах 5000-6000. Результат выполнения представлен на рисунке 13.

Реляционная алгебра

ScholarFiltered ← σ\_{name\_semester = '2023О' ∧ size\_scholarshp ≥ 5000 ∧ size\_scholarshp ≤ 6000}(scholarship)

Join ← ScholarFiltered ⋈\_{scholarship.id\_num\_student = student.id\_num\_student} student

Result ← π\_{first\_name\_student, second\_name\_student, surname\_student, size\_scholarshp}(Join)

SQL скрипт

SELECT s.first\_name\_student, s.second\_name\_student,

s.surname\_student, st.size\_scholarshp

FROM student s,scholarship st

WHERE st.id\_num\_student=s.id\_num\_student

and st.name\_semester='2023О'

and CAST(st.size\_scholarshp as numeric) BETWEEN 5000 and 6000;

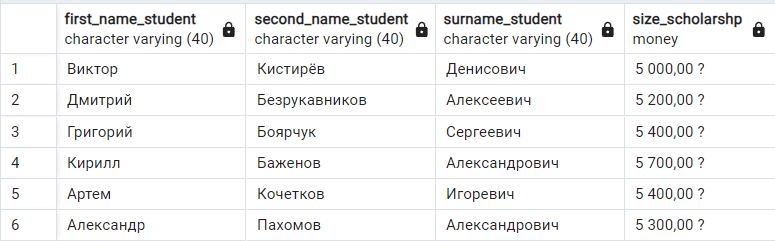


Рисунок 13 – Результат выполнения запроса

Запрос №5

Получение оценок студентов в том числе без оценок за какой-либо семестр. Результат выполнения представлен на рисунке 14.

Реляционная алгебра

Join ← student ⟕\_{student.id\_num\_student = mark.id\_num\_student} mark

Result ← π\_{first\_name\_student, second\_name\_student, surname\_student, lesson\_name\_mark, score\_mark}(Join)

Sorted ← τ\_{lesson\_name\_mark}(Result)

SQL скрипт

SELECT s.first\_name\_student, s.second\_name\_student,

s.surname\_student,m.lesson\_name\_mark, m.score\_mark

FROM student s

LEFT JOIN mark m ON s.id\_num\_student=m.id\_num\_student

ORDER BY m.lesson\_name\_mark

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 14 – Результат выполнения запроса

## Пользовательские представления

### Представление №1

Получить список студентов с указанием их оценок по всем предметам за заданный семестр. Форма выходных данных представлена в таблице 5.

Таблица 5 – форма выходных данных пользовательского представления

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id\_num\_student | Фамилия | Имя | Отчество | Предмет | Оценка | Тип оценки |

Входные параметры:

1. semester\_name — код семестра, тип данных – varchar, например '2023В'

Формулировка запроса на языке SQL:

CREATE OR REPLACE VIEW student\_marks\_by\_semester AS

SELECT

s.id\_num\_student,

s.second\_name\_student,

s.first\_name\_student,

s.surname\_student,

m.lesson\_name\_mark,

m.score\_mark,

m.type\_mark,

m.name\_semester

FROM

student s

JOIN

mark m ON s.id\_num\_student = m.id\_num\_student;

Пример вызова

SELECT \*

FROM student\_marks\_by\_semester

WHERE name\_semester = '2023В';

Результат выполнения представлен на рисунке 15.

### Представление №2

Показать студентов, которые получали стипендию в заданном семестре и в заданном диапазоне значений. Форма выходных данных представлена в таблице 6.

Таблица 6 – форма выходных данных пользовательского представления

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Фамилия** | **Имя** | **Отчество** | **Сумма стипендии** | **Семестр** |

Входные параметры

1. semester\_name — код семестра (например, '2023В')
2. min\_amount — минимальная сумма (например, 5000)
3. max\_amount — максимальная сумма (например, 6000)

Формулировка SQL запроса:

CREATE OR REPLACE VIEW student\_scholarship\_range AS

SELECT

s.second\_name\_student,

s.first\_name\_student,

s.surname\_student,

sc.size\_scholarshp,

sc.name\_semester

FROM

student s

JOIN

scholarship sc ON s.id\_num\_student = sc.id\_num\_student;

Пример выполнения:

SELECT \*

FROM student\_scholarship\_range

WHERE name\_semester = '2023В'

AND size\_scholarshp BETWEEN 5000 AND 6000;

Результат выполнения представлен на рисунке 16.

## Архитектура информационной системы

### Диаграмма компонентов

Разрабатываемая АИС имеет четырехуровневую клиент-серверную архитектуру, представленную на рисунке 17.

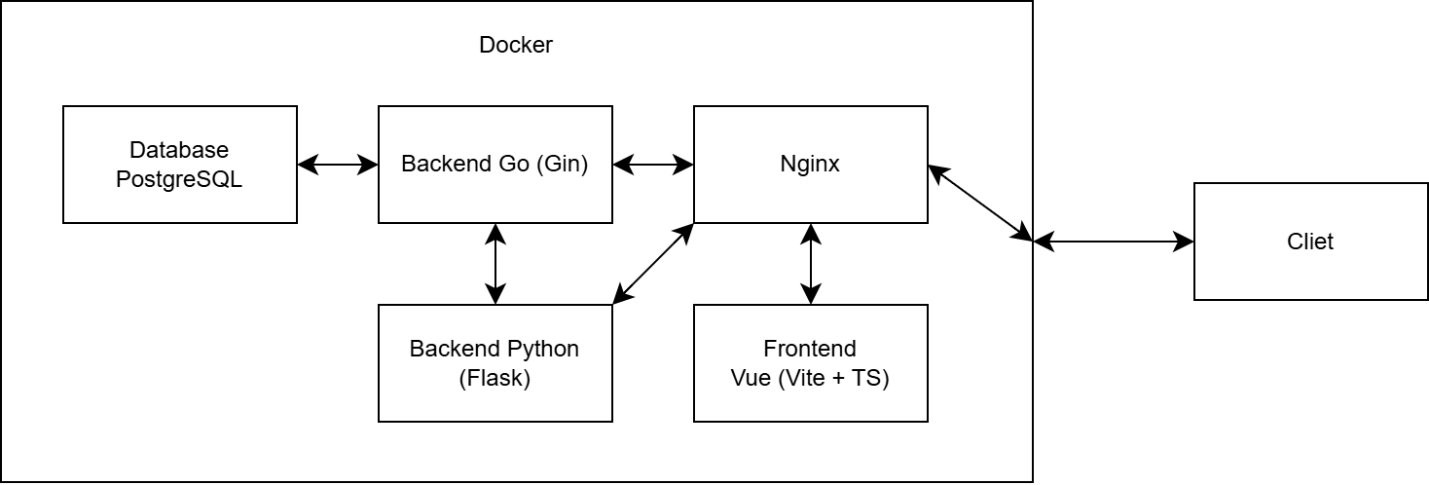


Рисунок 15 – Архитектура информационной системы

### Спецификация компонентов

База данных:

СУБД PostgreSQL 16

Backend:

Python 1.13.2 Flask

Go 1.24.0 Gin, pgx

Frontend:

Html, Css, TypeScript, Vue, Vite

Система оркестрации и контейнеризации:

Docker

Реверс-прокси сервер:

Nginx

### Распределение бизнес-логики между компонентами

Распределение бизнес логики между компонентами:

1. СУБД – выполнение запросов, обработка транзакций, хранение и управление данными, хранимые процедуры.
2. Backend
   1. Python Flask – Генерация отчетов PDF файлов по данным полученным с второго компонента Go Gin
   2. Go Gin – Обработка поступающих данных, подготовка и формирование запросов к СУБД, генерация ответов JSON.
3. Frontend – Интерфейс для работы пользователя, вызов серверных функций приложения

### Интерфейсы взаимодействия компонентов

Backend на Go GIN предоставляет интерфейсы взаимодействия с базой данных посредством GET, POST, PUT, DELETE методов HTTP-протокола через обратный прокси Nginx. Также с ним работает модуль Python Flask который предоставляет возможность генерировать отчеты. Серверная часть взаимодействует с базой данных при помощи драйвера pgx.

## Хранимые процедуры и триггеры

### Хранимая процедура №1

1. Процедура принимает на вход:
   * Порог среднего балла (threshold\_score).
   * Название семестра (current\_semester).
   * Тип бюджета (budget\_type).
2. Находит id\_budget из таблицы budget, соответствующий текущему семестру и указанному типу бюджета.
3. Для студентов, чей средний балл за текущий семестр превышает порог:
   * Добавляет записи о стипендии в таблицу scholarship.

На рисунке 18 представлена блок-схема алгоритма процедуры.

Изображение выглядит как текст, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 16 – Блок-схема транзакции

CREATE OR REPLACE PROCEDURE assign\_scholarships(current\_semester VARCHAR, budget\_type VARCHAR)

LANGUAGE plpgsql

AS $$

DECLARE

target\_budget\_id INTEGER;

BEGIN

SELECT id\_budget

INTO target\_budget\_id

FROM budget

WHERE name\_semester = current\_semester AND type\_scholarship\_budget = budget\_type

LIMIT 1;

IF target\_budget\_id IS NULL THEN

RAISE EXCEPTION 'Бюджет не найден для семестра % и типа %', current\_semester, budget\_type;

END IF;

DELETE FROM scholarship

WHERE name\_semester = current\_semester;

INSERT INTO scholarship (id\_num\_student, name\_semester, size\_scholarship, id\_budget)

SELECT

m.id\_num\_student,

current\_semester,

4500+(AVG(m.score\_mark)-80)\*100,

target\_budget\_id

FROM

mark m

WHERE

m.name\_semester = current\_semester

GROUP BY

m.id\_num\_student, m.name\_semester

HAVING

AVG(m.score\_mark) >= 80;

RAISE NOTICE 'Стипендии начислены студентам за семестр: % и тип бюджета: %', current\_semester, budget\_type;

END;

$$;

На рисунке 19 представлено содержимое таблицы scholarship до выполнения хранимой процедуры.

Изображение выглядит как текст, число, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 17 – Таблица scholarship до вызова процедуры

На рисунке 20 представлено содержимое таблицы scholarship после выполнения хранимой процедуры.

Изображение выглядит как текст, число, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 18 – Таблица scholarship после вызова процедуры

### Триггер №1

Триггер проверяет корректность добавляемой оценки в таблицу mark. Если значение score\_mark выходит за пределы допустимого диапазона (0–100), выполнение операции прерывается, а добавление записи отменяется.

SQL скрипт создания триггера

CREATE OR REPLACE FUNCTION check\_mark\_score()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.score\_mark < 0 OR NEW.score\_mark > 100 THEN

RAISE EXCEPTION 'Оценка должна быть в диапазоне от 0 до 100. Некорректное значение: %', NEW.score\_mark;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_check\_mark\_score

BEFORE INSERT OR UPDATE ON mark

FOR EACH ROW

EXECUTE FUNCTION check\_mark\_score();

Пример выполнения

INSERT INTO mark (id\_num\_student, name\_semester, lesson\_name\_mark, score\_mark, type\_mark)

VALUES (1220060405, '2023В', 'Физика', 150, 'Экзамен');

На рисунке 21 представлена таблица после выполнения запроса.

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 19 – Пример выполнения триггера

Триггер №2

Триггер срабатывает перед вставкой записи в таблицу mark. Он автоматически присваивает значение в поле type\_mark в зависимости от поля score\_mark.

SQL скрипт создания триггера

CREATE OR REPLACE FUNCTION set\_type\_mark()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.score\_mark >= 93 THEN

NEW.type\_mark := 'Отлично';

ELSIF NEW.score\_mark >= 80 THEN

NEW.type\_mark := 'Хорошо';

ELSIF NEW.score\_mark >= 53 THEN

NEW.type\_mark := 'Удовлетворительно';

ELSE

NEW.type\_mark := 'Неудовлетворительно';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger\_set\_type\_mark

BEFORE INSERT ON mark

FOR EACH ROW

EXECUTE FUNCTION set\_type\_mark();

Запрос для добавления оценки представлен ниже.

INSERT INTO mark (id\_num\_student, name\_semester, lesson\_name\_mark, score\_mark, type\_mark)

VALUES (1220060405, '2023В', 'Физика', 82, 'Экзамен');

На рисунке 22 представлена таблица после выполнения запроса.

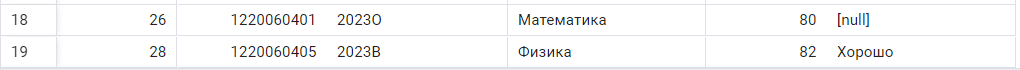


Рисунок 20 – Пример выполнения триггера

# Рабочий проект

## SQL-скрипт создания структуры БД

Разработанный SQL-скрипт предназначенный для создания структуры БД рабочего проекта представлен в приложении А.

## SQL-скрипт триггеров и хранимых процедур

Разработанные скрипты триггеров, а также хранимая процедура представлены в приложении Б.

## Текст программы

https://github.com/LiveisFpv/Go-web

## Руководство пользователя

Заключение

Приложение А

BEGIN;

CREATE TABLE IF NOT EXISTS public.achievement

(

id\_achivment serial NOT NULL,

id\_num\_student integer NOT NULL,

id\_category integer NOT NULL,

name\_achivement character varying(200) COLLATE pg\_catalog."default" NOT NULL,

date\_achivment date NOT NULL,

CONSTRAINT pk\_achievement PRIMARY KEY (id\_achivment)

);

CREATE TABLE IF NOT EXISTS public.budget

(

type\_scholarship\_budget character varying(100) COLLATE pg\_catalog."default" NOT NULL,

name\_semester character varying(5) COLLATE pg\_catalog."default" NOT NULL,

size\_budget numeric NOT NULL,

id\_budget serial NOT NULL,

CONSTRAINT pk\_budget PRIMARY KEY (id\_budget)

);

CREATE TABLE IF NOT EXISTS public.category

(

id\_category serial NOT NULL,

achivments\_type\_category character varying(50) COLLATE pg\_catalog."default" NOT NULL,

score\_category smallint NOT NULL,

CONSTRAINT pk\_category PRIMARY KEY (id\_category)

);

CREATE TABLE IF NOT EXISTS public."group"

(

name\_group character varying(10) COLLATE pg\_catalog."default" NOT NULL,

studies\_direction\_group character varying(8) COLLATE pg\_catalog."default" NOT NULL,

studies\_profile\_group character varying(50) COLLATE pg\_catalog."default" NOT NULL,

start\_date\_group date NOT NULL,

studies\_period\_group smallint NOT NULL,

CONSTRAINT pk\_group PRIMARY KEY (name\_group)

);

CREATE TABLE IF NOT EXISTS public.mark

(

id\_mark serial NOT NULL,

id\_num\_student integer NOT NULL,

name\_semester character varying(5) COLLATE pg\_catalog."default" NOT NULL,

lesson\_name\_mark character varying(100) COLLATE pg\_catalog."default" NOT NULL,

score\_mark smallint NOT NULL,

type\_mark character varying(20) COLLATE pg\_catalog."default",

type\_exam character varying(20) COLLATE pg\_catalog."default" NOT NULL DEFAULT 'Семестр'::character varying,

CONSTRAINT pk\_mark PRIMARY KEY (id\_mark)

);

CREATE TABLE IF NOT EXISTS public.scholarship

(

id\_scholarship serial NOT NULL,

id\_num\_student integer NOT NULL,

name\_semester character varying(5) COLLATE pg\_catalog."default" NOT NULL,

size\_scholarshp numeric NOT NULL,

id\_budget integer NOT NULL,

CONSTRAINT pk\_scholarship PRIMARY KEY (id\_scholarship)

);

CREATE TABLE IF NOT EXISTS public.semester

(

name\_semester character varying(5) COLLATE pg\_catalog."default" NOT NULL,

date\_start\_semester date NOT NULL,

date\_end\_semester date NOT NULL,

CONSTRAINT pk\_semester PRIMARY KEY (name\_semester)

);

CREATE TABLE IF NOT EXISTS public.student

(

id\_num\_student integer NOT NULL,

name\_group character varying(10) COLLATE pg\_catalog."default" NOT NULL,

email\_student character varying(40) COLLATE pg\_catalog."default" NOT NULL,

second\_name\_student character varying(40) COLLATE pg\_catalog."default" NOT NULL,

first\_name\_student character varying(40) COLLATE pg\_catalog."default" NOT NULL,

surname\_student character varying(40) COLLATE pg\_catalog."default",

CONSTRAINT pk\_student PRIMARY KEY (id\_num\_student),

CONSTRAINT ak\_email\_student\_student UNIQUE (email\_student)

);

CREATE TABLE IF NOT EXISTS public."user"

(

user\_id serial NOT NULL,

user\_login character varying(40) COLLATE pg\_catalog."default" NOT NULL,

user\_email character varying(40) COLLATE pg\_catalog."default" NOT NULL,

user\_student\_id integer,

user\_role character varying(20) COLLATE pg\_catalog."default" NOT NULL DEFAULT 'USER'::character varying,

user\_password text COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT user\_pkey PRIMARY KEY (user\_id),

CONSTRAINT user\_user\_email\_key UNIQUE (user\_email),

CONSTRAINT user\_user\_login\_key UNIQUE (user\_login)

);

ALTER TABLE IF EXISTS public.achievement

ADD CONSTRAINT fk\_achievem\_have\_student FOREIGN KEY (id\_num\_student)

REFERENCES public.student (id\_num\_student) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE CASCADE;

CREATE INDEX IF NOT EXISTS have\_fk

ON public.achievement(id\_num\_student);

ALTER TABLE IF EXISTS public.achievement

ADD CONSTRAINT fk\_achievem\_refer\_category FOREIGN KEY (id\_category)

REFERENCES public.category (id\_category) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS refer\_fk

ON public.achievement(id\_category);

ALTER TABLE IF EXISTS public.mark

ADD CONSTRAINT fk\_mark\_get\_student FOREIGN KEY (id\_num\_student)

REFERENCES public.student (id\_num\_student) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE CASCADE;

CREATE INDEX IF NOT EXISTS get\_fk

ON public.mark(id\_num\_student);

ALTER TABLE IF EXISTS public.mark

ADD CONSTRAINT fk\_mark\_give\_semester FOREIGN KEY (name\_semester)

REFERENCES public.semester (name\_semester) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS give\_fk

ON public.mark(name\_semester);

ALTER TABLE IF EXISTS public.scholarship

ADD CONSTRAINT fk\_scholars\_assign\_semester FOREIGN KEY (name\_semester)

REFERENCES public.semester (name\_semester) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS assign\_fk

ON public.scholarship(name\_semester);

ALTER TABLE IF EXISTS public.scholarship

ADD CONSTRAINT fk\_scholars\_calculate\_budget FOREIGN KEY (id\_budget)

REFERENCES public.budget (id\_budget) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE RESTRICT;

ALTER TABLE IF EXISTS public.scholarship

ADD CONSTRAINT fk\_scholars\_receive\_student FOREIGN KEY (id\_num\_student)

REFERENCES public.student (id\_num\_student) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE CASCADE;

CREATE INDEX IF NOT EXISTS receive\_fk

ON public.scholarship(id\_num\_student);

ALTER TABLE IF EXISTS public.student

ADD CONSTRAINT fk\_student\_study\_group FOREIGN KEY (name\_group)

REFERENCES public."group" (name\_group) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE RESTRICT;

CREATE INDEX IF NOT EXISTS study\_fk

ON public.student(name\_group);

END;

Приложение Б

(SQL-скрипт триггеров и хранимых процедур)

Хранимая процедура

CREATE OR REPLACE PROCEDURE assign\_scholarships(current\_semester VARCHAR, budget\_type VARCHAR)

LANGUAGE plpgsql

AS $$

DECLARE

target\_budget\_id INTEGER;

BEGIN

SELECT id\_budget

INTO target\_budget\_id

FROM budget

WHERE name\_semester = current\_semester AND type\_scholarship\_budget = budget\_type

LIMIT 1;

IF target\_budget\_id IS NULL THEN

RAISE EXCEPTION 'Бюджет не найден для семестра % и типа %', current\_semester, budget\_type;

END IF;

DELETE FROM scholarship

WHERE name\_semester = current\_semester;

INSERT INTO scholarship (id\_num\_student, name\_semester, size\_scholarship, id\_budget)

SELECT

m.id\_num\_student,

current\_semester,

4500+(AVG(m.score\_mark)-80)\*100,

target\_budget\_id

FROM

mark m

WHERE

m.name\_semester = current\_semester

GROUP BY

m.id\_num\_student, m.name\_semester

HAVING

AVG(m.score\_mark) >= 80;

RAISE NOTICE 'Стипендии начислены студентам за семестр: % и тип бюджета: %', current\_semester, budget\_type;

END;

$$;

Триггер 1

CREATE OR REPLACE FUNCTION check\_mark\_score()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.score\_mark < 0 OR NEW.score\_mark > 100 THEN

RAISE EXCEPTION 'Оценка должна быть в диапазоне от 0 до 100. Некорректное значение: %', NEW.score\_mark;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_check\_mark\_score

BEFORE INSERT OR UPDATE ON mark

FOR EACH ROW

EXECUTE FUNCTION check\_mark\_score();

Триггер 2

SQL скрипт создания триггера

CREATE OR REPLACE FUNCTION set\_type\_mark()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.score\_mark >= 93 THEN

NEW.type\_mark := 'Отлично';

ELSIF NEW.score\_mark >= 80 THEN

NEW.type\_mark := 'Хорошо';

ELSIF NEW.score\_mark >= 53 THEN

NEW.type\_mark := 'Удовлетворительно';

ELSE

NEW.type\_mark := 'Неудовлетворительно';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger\_set\_type\_mark

BEFORE INSERT ON mark

FOR EACH ROW

EXECUTE FUNCTION set\_type\_mark();

Приложение В

(Руководство пользователя)

**1 Введение**

**2 Назначения и условия применения**

**3 Подготовка к работе**

**4 Описание операций**

**5 Аварийные ситуации**

**6 Рекомендации к освоению**