

**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

\_\_\_\_\_

подпись, дата

Алексеев В.А.

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
Введение .....	6
1 Техническое задание .....	7
1.1 Характеристика предметной области .....	7
1.2 Цели и задачи системы .....	8
1.3 Автоматизируемые бизнес-процессы.....	9
1.4 Характеристика пользователей .....	10
1.5 Модель вариантов использования .....	11
1.6 Модели бизнес-процессов в нотации BPMN .....	12
1.7 Требования к информационному обеспечению .....	13
1.8 Требования к программному обеспечению и пользовательскому интерфейсу	14
2 Технический проект.....	16
2.1 Модели локальных представлений .....	16
2.1.1 Локальные ER-диаграммы.....	16
2.1.2 Основные запросы .....	19
2.1.3 Общая ER-диаграмма .....	20
2.2 Концептуальная модель данных .....	21
2.2.1 ER-диаграмма .....	21
2.2.2 Спецификация сущностей .....	22
2.2.3 Спецификация связей .....	25
2.3 Логическая модель данных .....	26
2.3.1 Диаграмма логической модели .....	26
2.3.2 Спецификация логической модели .....	27

2.4	Физическая модель данных .....	31
2.4.1	Обоснование выбора СУБД .....	31
2.4.2	Диаграмма физической модели .....	32
2.4.3	Спецификация таблиц .....	33
2.4.4	Проектирование вторичных индексов .....	37
2.5	Проектирование запросов выборки .....	38
2.5.1	Запрос №1 .....	38
2.5.2	Запрос №2 .....	39
2.5.3	Запрос №3 .....	40
2.5.4	Запрос №4 .....	41
2.5.5	Запрос №5 .....	42
2.6	Пользовательские представления .....	43
2.6.1	Представление №1 .....	43
2.6.2	Представление №2 .....	45
2.7	Архитектура информационной системы .....	47
2.7.1	Диаграмма компонентов .....	47
2.7.2	Спецификация компонентов .....	48
2.7.3	Распределение бизнес-логики между компонентами .....	49
2.7.4	Интерфейсы взаимодействия компонентов .....	50
2.8	Хранимые процедуры и триггеры .....	51
2.8.1	Хранимая процедура №1 .....	51
2.8.2	Триггер №1 .....	55
2.8.3	Триггер №2 .....	57
3	Рабочий проект .....	59
3.1	SQL-скрипт создания структуры БД .....	59

3.2	SQL-скрипт триггеров и хранимых процедур .....	59
3.3	Текст программы .....	59
3.4	Руководство пользователя .....	59
	Заключение.....	60
	Приложение А.....	61
	Приложение Б .....	68
	Приложение В.....	71

## **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. Приложение 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**

### **1.1 Характеристика предметной области**

Предметная область 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

## **1.2 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 участниками 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.



### **1.3 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.

## **1.4 Характеристика пользователей**

### **1. Бухгалтерия**

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

### **2. 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.

### **3. Деканат**

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

## 1.5 Модель вариантов использования

Модель вариантов использования предоставлена на рисунке 1.

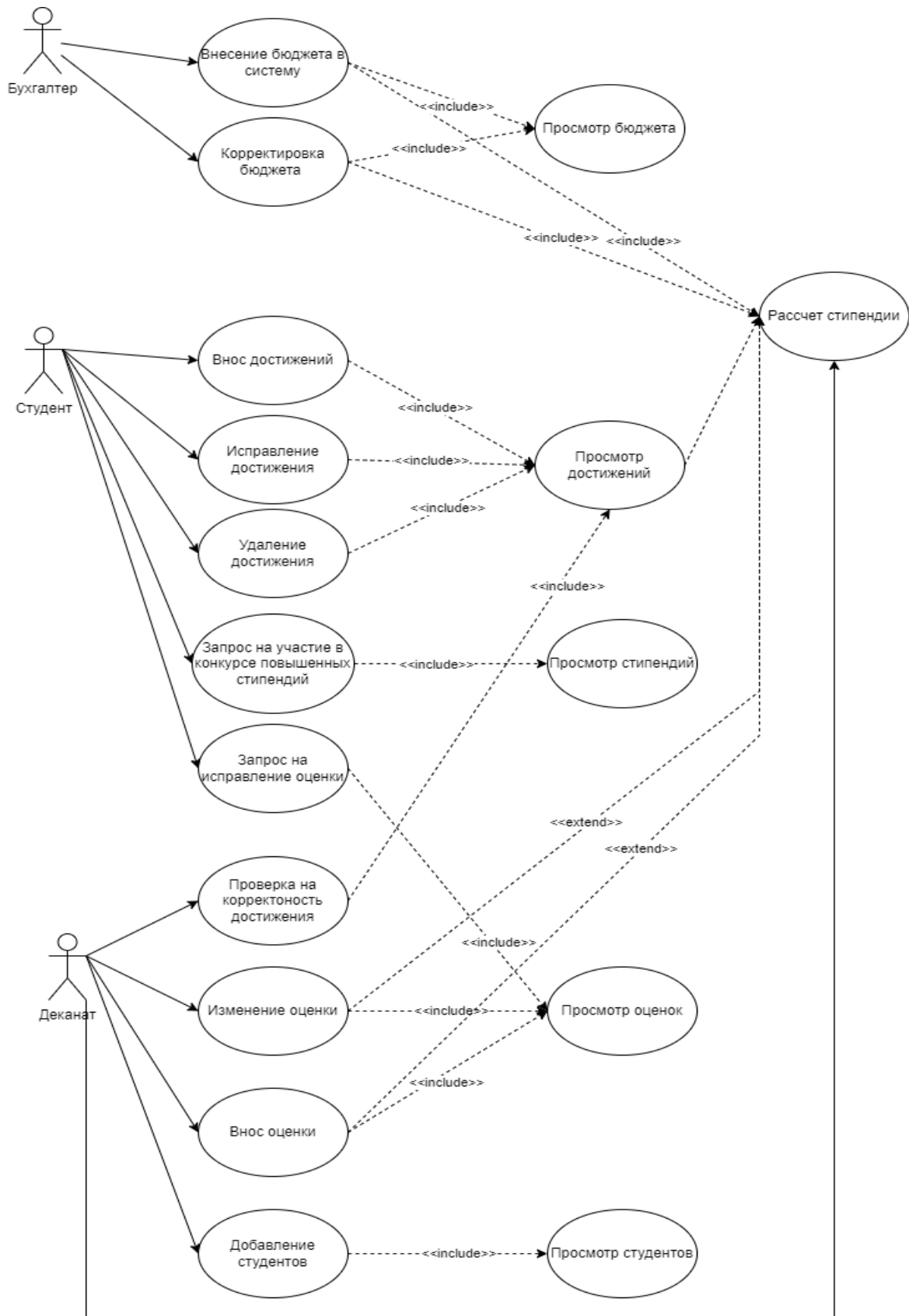


Рисунок 1 – use-case диаграмма

## 1.6 Модели бизнес-процессов в нотации BPMN

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

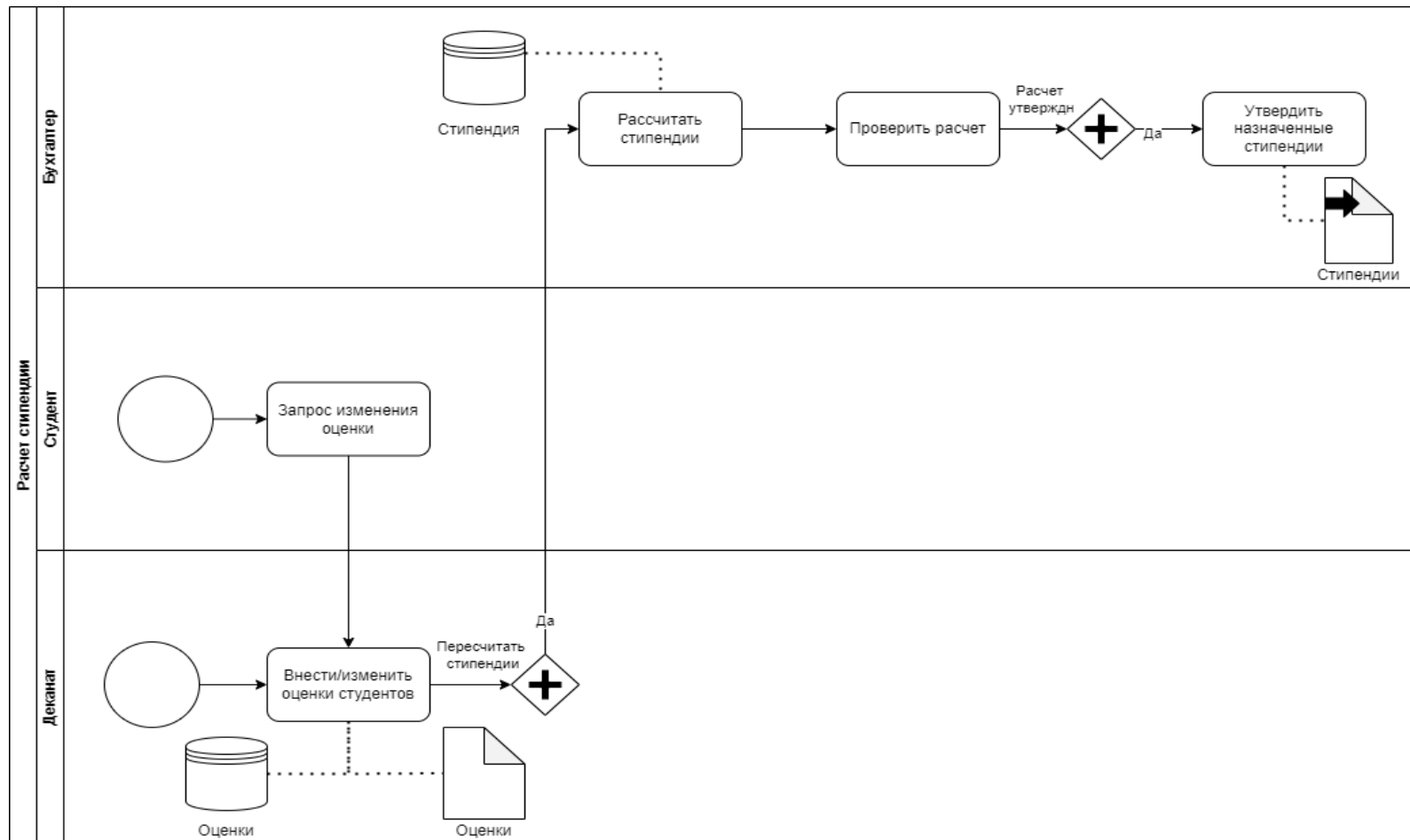


Рисунок 2 – модель BPMN бизнес-процесса

### **1.7 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 данным.

## **1.8 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.).

## 2 Технический проект

### 2.1 Модели локальных представлений

#### 2.1.1 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.

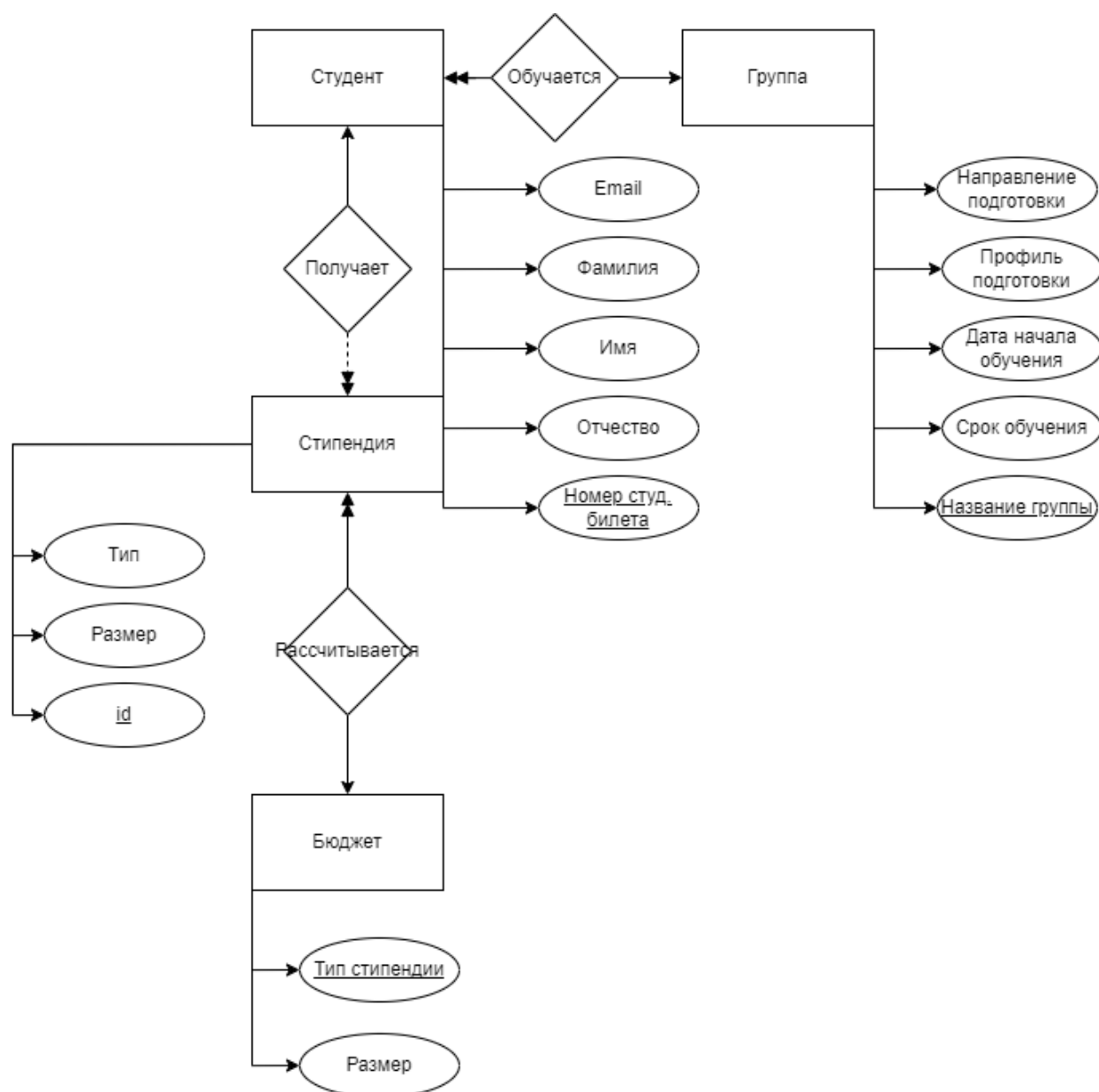


Рисунок 3 – Локальное представление со стороны бухгалтерии



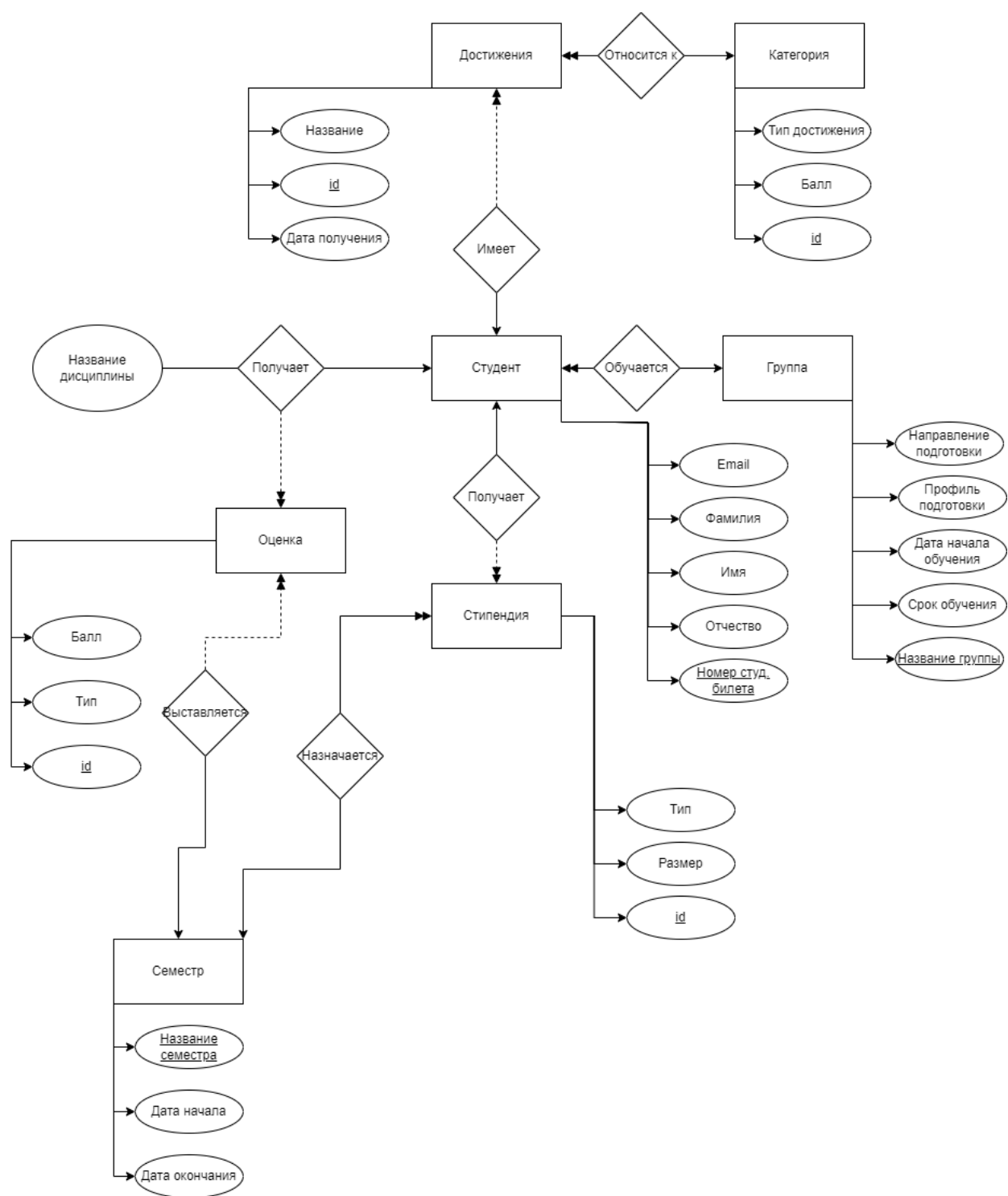


Рисунок 4 – Локальное представление со стороны студента

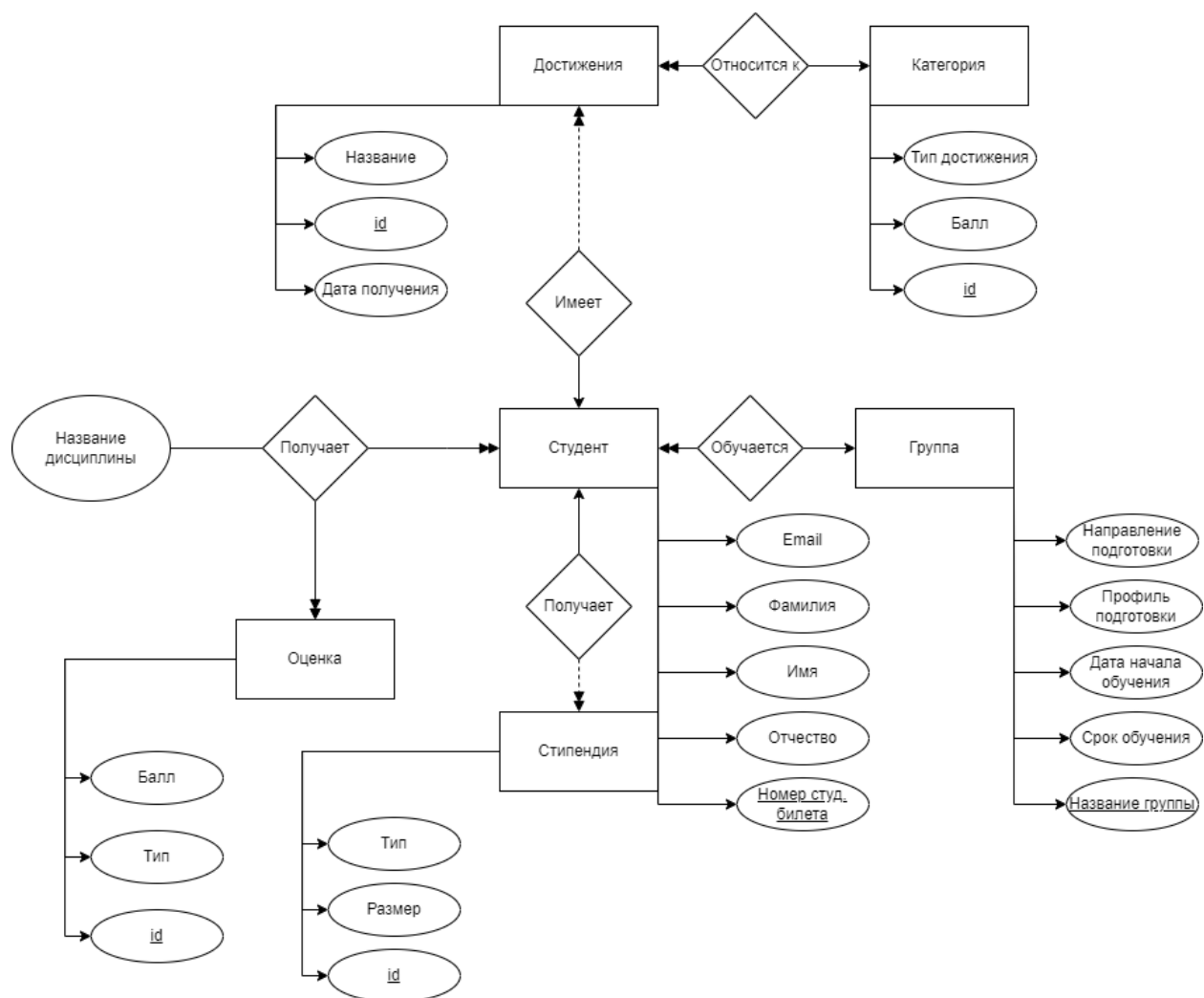


Рисунок 5 – Локальное представление со стороны студента

### **2.1.2 Основные запросы**

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

### 2.1.3 Общая ER-диаграмма

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

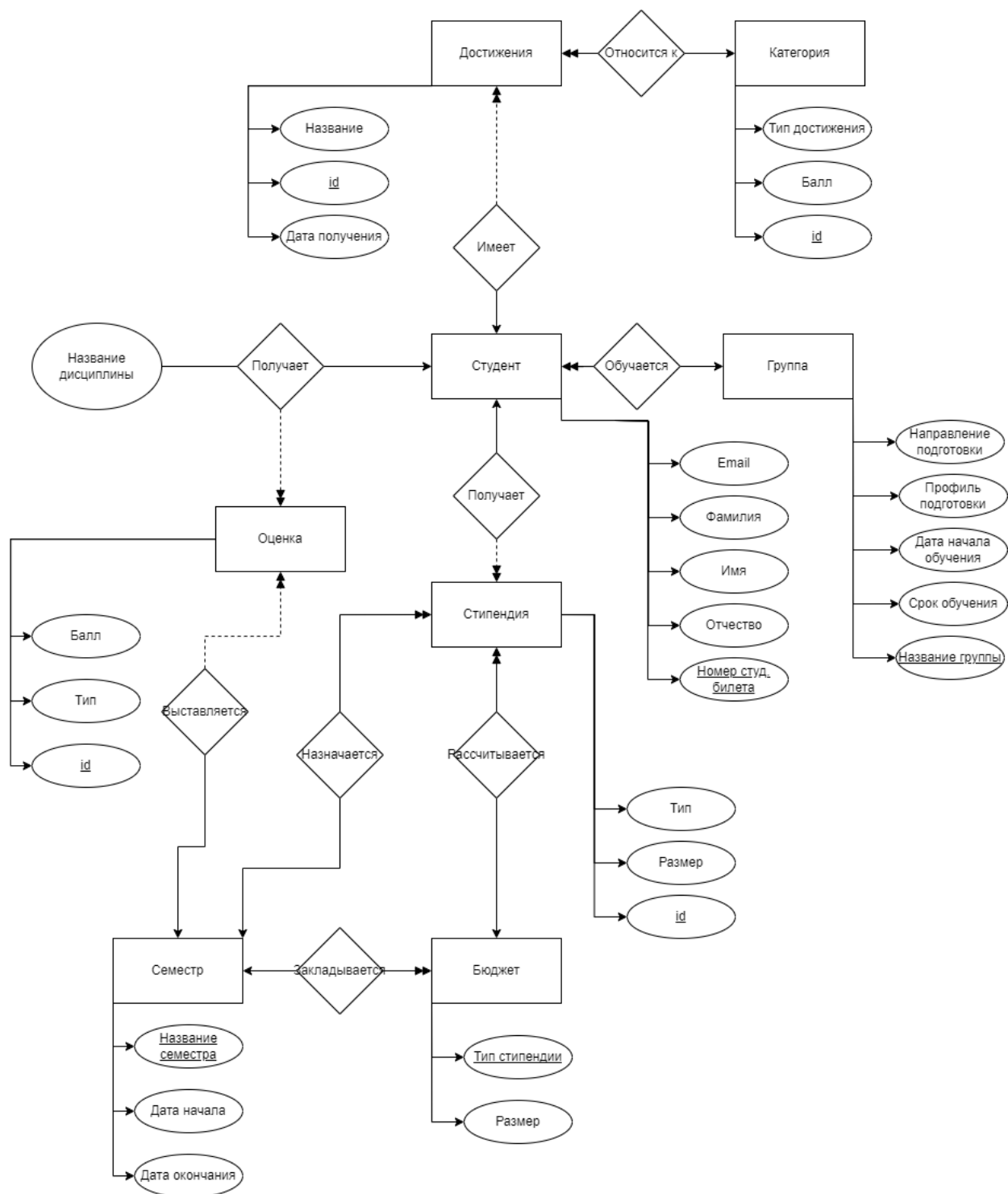


Рисунок 6 – Общая ER-диаграмма

## 2.2 Концептуальная модель данных

### 2.2.1 ER-диаграмма

На рисунках 7 и 8 представлена концептуальная модель данных в нотации Crow's Foot .

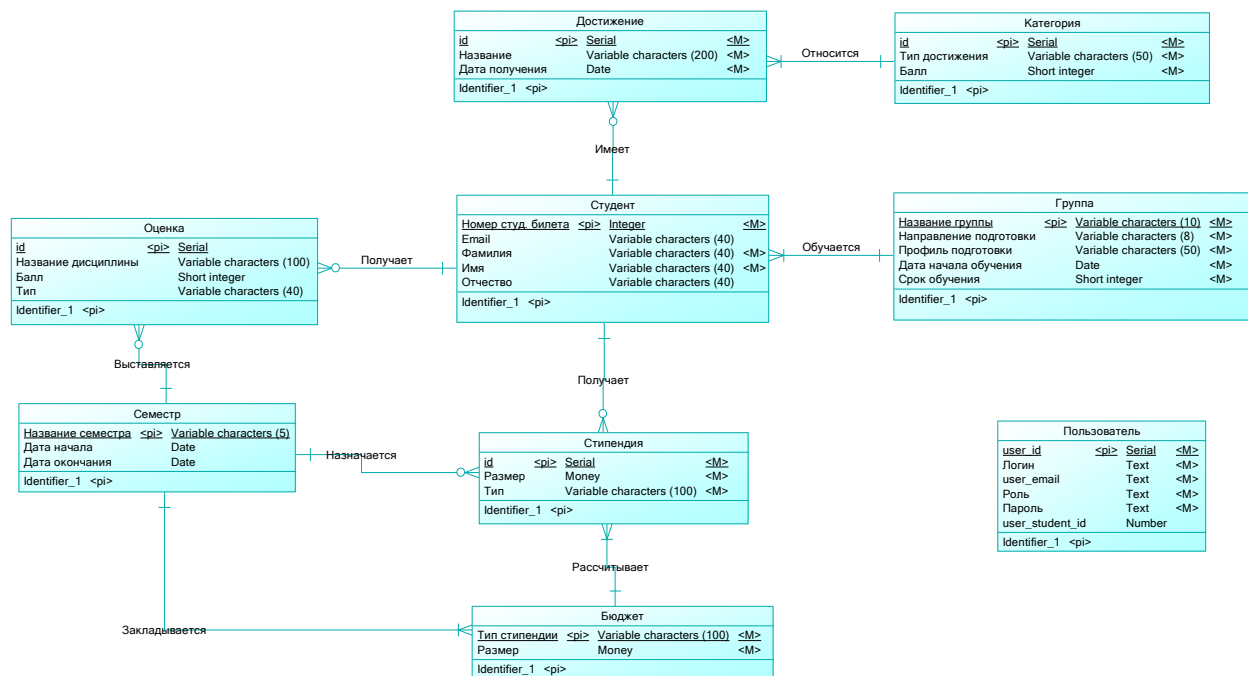


Рисунок 7 – Концептуальная модель данных с отображением name

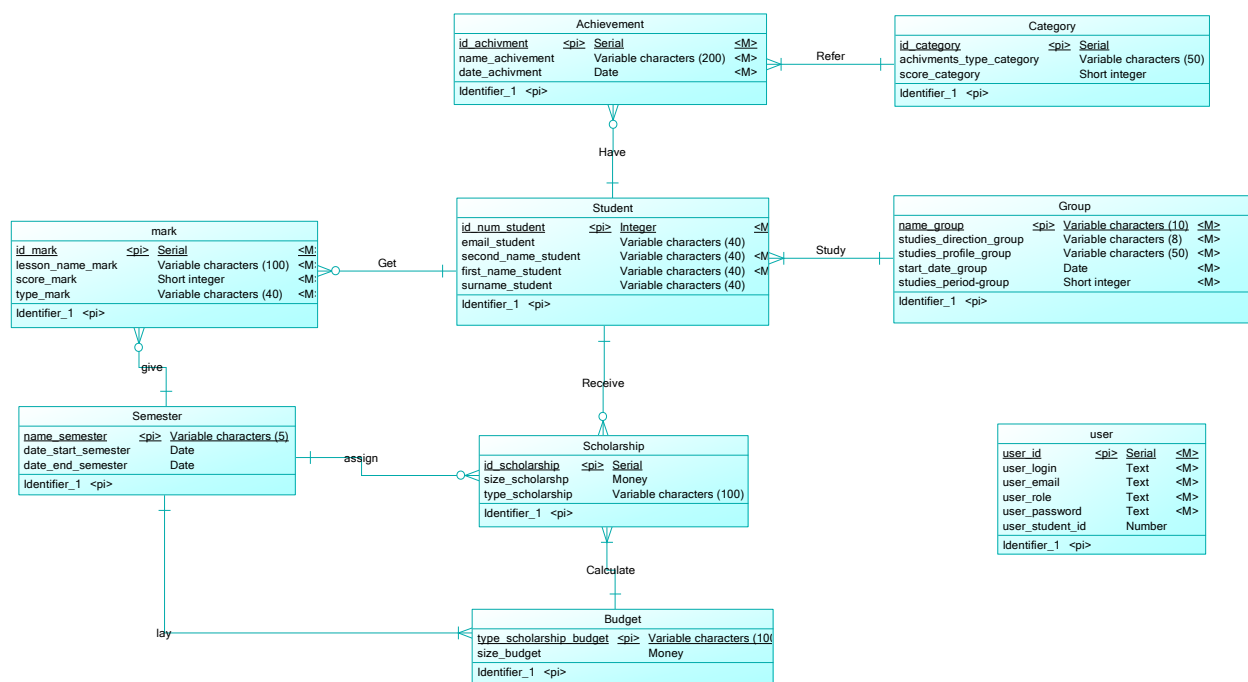


Рисунок 8 – Концептуальная модель данных с отображением code

### 2.2.2 Спецификация сущностей

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



### 2.2.3 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

## 2.3 Логическая модель данных

### 2.3.1 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 9 и 10.

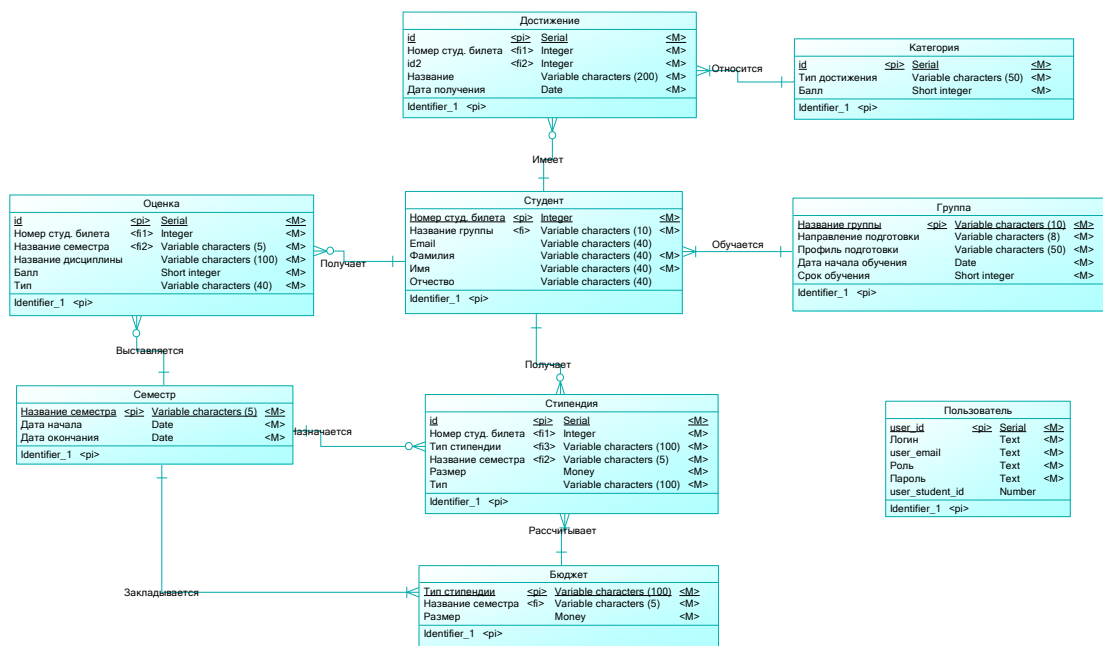


Рисунок 9 – Логическая модель в нотации Craw's Foot с обозначением name

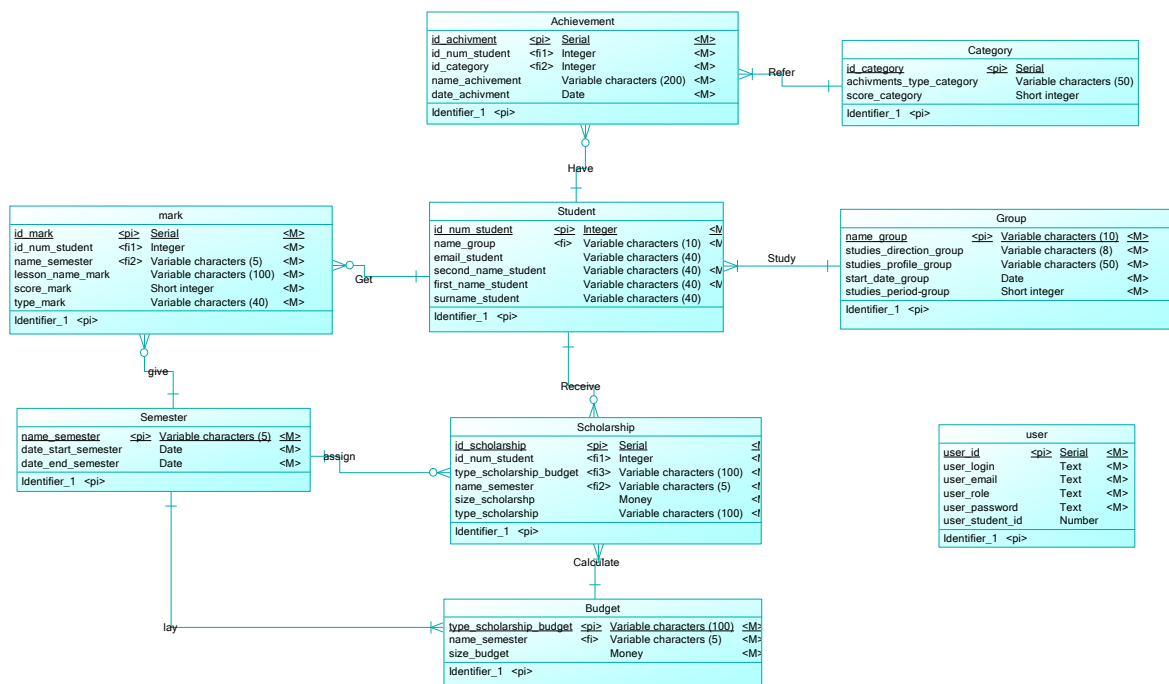


Рисунок 10 – Логическая модель в нотации Craw's Foot с обозначением code

### 2.3.2 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_semeste r	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_stu dent	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	-	-	

## **2.4 Физическая модель данных**

### **2.4.1 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.

## 2.4.2 Диаграмма физической модели

На основе логической модели, изображенной на рисунке 9, находящемся в пункте 2.3.1, построена физическая модель данных в нотации Craw`s Foot. Физическая модель данных изображена на рисунке 11.

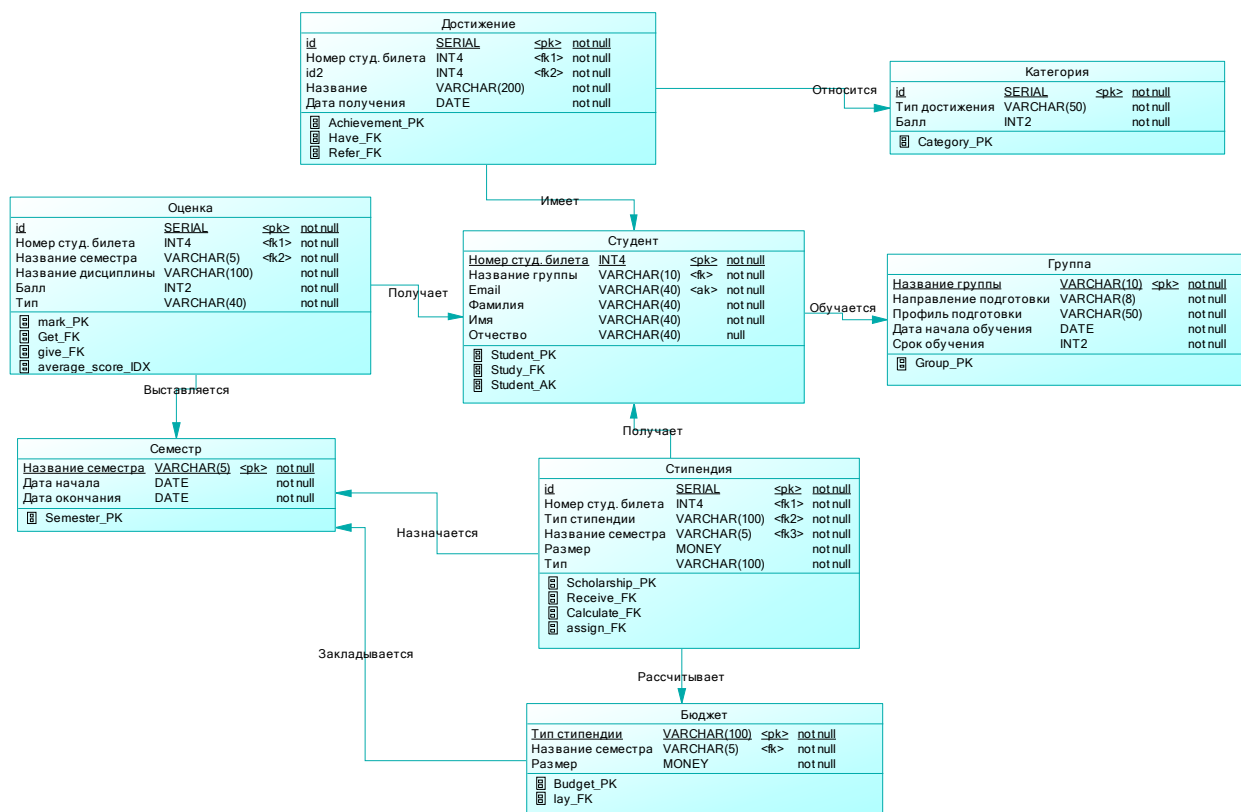


Рисунок 11 – Физическая модель данных для СУБД PostgreSQL



### 2.4.3 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_achivment	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_scholarship	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	-	-	

#### **2.4.4 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  
);
```

## 2.5 Проектирование запросов выборки

### 2.5.1 Запрос №1

Получение среднего балла студента с `id_num_student = 1220060404` и сравнение его с фиксированным значением. Результат выполнения представлен на рисунке 12.

Реляционная алгебра

$\text{Student1220} \leftarrow \sigma_{\{\text{id\_num\_student} = 1220060404\}}(\text{student})$

$\text{AvgMark1220} \leftarrow \gamma_{\{\text{id\_num\_student}; \text{AVG}(\text{score\_mark}) \rightarrow \text{avg\_mark}\}}(\sigma_{\{\text{id\_num\_student} = 1220060404\}}(\text{mark}))$

$\text{Result} \leftarrow \text{Student1220} \bowtie \text{AvgMark1220}$

$\pi_{\{\text{first\_name\_student}, \text{second\_name\_student}, \text{avg\_mark}\}}(\text{Result})$

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;

	first_name_student character varying (40) 🔒	second_name_student character varying (40) 🔒	avg_mark numeric 🔒
1	Виктор	Кистирёв	77.2500000000000000

Рисунок 12 – Результат выполнения запроса

### 2.5.2 Запрос №2

Количество студентов в каждой группе. Результат выполнения представлен на рисунке 13.

Реляционная алгебра

$\gamma_{\{\text{name\_group}; \text{COUNT}(\ast) \rightarrow \text{students\_count}\}}(\text{student})$

SQL скрипты

```
SELECT name_group, COUNT(*) AS students_count
```

```
FROM student
```

```
GROUP BY name_group;
```

	<b>name_group</b> character varying (10) 🔒	<b>students_count</b> bigint 🔒
1	АС-22-1	8
2	АИ-22	5
3	ПИ-22-1	11

Рисунок 13– Результат выполнения запроса

### 2.5.3 Запрос №3

Выбрать студентов, у которых есть хотя бы одна оценка выше 85.  
Результат выполнения представлен на рисунке 14.

Реляционная алгебра

Result  $\leftarrow$  student  $\bowtie_{\{ \text{student.id\_num\_student} = \text{mark.id\_num\_student} \wedge \text{mark.score\_mark} > 85 \}}$  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

);

	id_num_student [PK] integer	name_group character varying (10)	email_student character varying (40)	second_name_student character varying (40)	first_name_student character varying (40)	surname_student character varying (40)
1	1220060407	ПИ-22-1	ase@gmail.com	Пахомов	Александр	Александрович
2	1220060405	ПИ-22-1	qse@gmail.com	Кочетков	Артем	Игоревич
3	1220060404	ПИ-22-1	qwd@gmail.com	Кистирёв	Виктор	Денисович
4	1220060402	ПИ-22-1	asd@gmail.com	Боярчук	Григорий	Сергеевич
5	1220060401	ПИ-22-1	qwe@gmail.com	Баженов	Кирилл	Александрович

Рисунок 14 – Результат выполнения запроса



#### 2.5.4 Запрос №4

Получение студентов за определённый семестр со стипендией в пределах 5000-6000. Результат выполнения представлен на рисунке 15.

Реляционная алгебра

$$\text{ScholarFiltered} \leftarrow \sigma_{\{\text{name\_semester} = '2023O' \wedge \text{size\_scholarshp} \geq 5000 \wedge \text{size\_scholarshp} \leq 6000\}}(\text{scholarship})$$
$$\text{Join} \leftarrow \text{ScholarFiltered} \bowtie_{\{\text{scholarship.id\_num\_student} = \text{student.id\_num\_student}\}} \text{student}$$
$$\text{Result} \leftarrow \pi_{\{\text{first\_name\_student}, \text{second\_name\_student}, \text{surname\_student}, \text{size\_scholarshp}\}}(\text{Join})$$

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='2023O'
```

```
and CAST(st.size_scholarshp as numeric) BETWEEN 5000 and 6000;
```

	first_name_student character varying (40) 🔒	second_name_student character varying (40) 🔒	surname_student character varying (40) 🔒	size_scholarshp money 🔒
1	Виктор	Кистирёв	Денисович	5 000,00 ?
2	Дмитрий	Безрукавников	Алексеевич	5 200,00 ?
3	Григорий	Боярчук	Сергеевич	5 400,00 ?
4	Кирилл	Баженов	Александрович	5 700,00 ?
5	Артём	Кочетков	Игоревич	5 400,00 ?
6	Александр	Пахомов	Александрович	5 300,00 ?

Рисунок 15 – Результат выполнения запроса

### 2.5.5 Запрос №5

Получение оценок студентов в том числе без оценок за какой-либо семестр. Результат выполнения представлен на рисунке 16.

Реляционная алгебра

$\text{Join} \leftarrow \text{student} \bowtie_{\{\text{student.id\_num\_student} = \text{mark.id\_num\_student}\}} \text{mark}$

$\text{Result} \leftarrow \pi_{\{\text{first\_name\_student}, \text{second\_name\_student}, \text{surname\_student}, \text{lesson\_name\_mark}, \text{score\_mark}\}}(\text{Join})$

$\text{Sorted} \leftarrow \tau_{\{\text{lesson\_name\_mark}\}}(\text{Result})$

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
```

	first_name_student character varying (40) 🔒	second_name_student character varying (40) 🔒	surname_student character varying (40) 🔒	lesson_name_mark character varying (100) 🔒	score_mark smallint 🔒
13	Григорий	Боярчук	Сергеевич	Математика	89
14	Григорий	Боярчук	Сергеевич	Математика	77
15	Виктор	Кистирёв	Денисович	Программирование	100
16	Виктор	Кистирёв	Денисович	Программирование	56
17	Виктор	Кистирёв	Денисович	Программирование	75
18	Виктор	Кистирёв	Денисович	Программирование	78
19	Иван	Синюков	Евгеньевич	[null]	[null]
20	Владимир	Теплов	Сергеевич	[null]	[null]
21	Тимофей	Синицкий	Максимович	[null]	[null]
22	Данила	Сазонов	Андреевич	[null]	[null]
23	Дмитрий	Обыденных	Александрович	[null]	[null]
24	Денис	Ульянкин	Сергеевич	[null]	[null]

Рисунок 16 – Результат выполнения запроса

## 2.6 Пользовательские представления

### 2.6.1 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 = '2023B';

Результат выполнения представлен на рисунке 17.

	id_num_student integer	second_name_student character varying (40)	first_name_student character varying (40)	surname_student character varying (40)	lesson_name_mark character varying (100)	score_mark smallint	type_mark character varying (20)	name_semester character varying (
1	1220060405	Кочетков	Артём	Игоревич	Базы данных	88	Хорошо	2023B
2	1220060405	Кочетков	Артём	Игоревич	Базы данных	89	Хорошо	2023B
3	1220060405	Кочетков	Артём	Игоревич	Физика	82	Хорошо	2023B
4	1220060404	Кистирёв	Виктор	Денисович	Программирование	100	Отлично	2023B
5	1220060404	Кистирёв	Виктор	Денисович	Программирование	75	Удовлетворительно	2023B
6	1220060402	Боярчук	Григорий	Сергеевич	Математика	56	Удовлетворительно	2023B
7	1220060402	Боярчук	Григорий	Сергеевич	Математика	66	Удовлетворительно	2023B
8	1220060401	Баженов	Кирилл	Александрович	Математика	99	Отлично	2023B
9	1220060401	Баженов	Кирилл	Александрович	Математика	76	Удовлетворительно	2023B

Рисунок 17 – Результат выполнения

## 2.6.2 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 18.

	second_name_student character varying (40) 🔒	first_name_student character varying (40) 🔒	surname_student character varying (40) 🔒	size_scholarshp numeric 🔒	name_semester character varying (5) 🔒
1	Бобенко	Максим	Геннадьевич	5700.00	2023В
2	Пахомов	Александр	Александрович	5300.00	2023В
3	Кочетков	Артем	Игоревич	5500.00	2023В
4	Кистирёв	Виктор	Денисович	5200.00	2023В
5	Боярчук	Григорий	Сергеевич	5300.00	2023В
6	Баженов	Кирилл	Александрович	5400.00	2023В

Рисунок 18 – Результат выполнения

## 2.7 Архитектура информационной системы

### 2.7.1 Диаграмма компонентов

Разрабатываемая АИС имеет четырехуровневую клиент-серверную архитектуру, представленную на рисунке 19. включающую в себя следующие уровни:

- Уровень хранения данных (Database),
- Серверный уровень обработки (Backend),
- Уровень веб-сервера (Reverse Proxy),
- Клиентский уровень (Frontend)

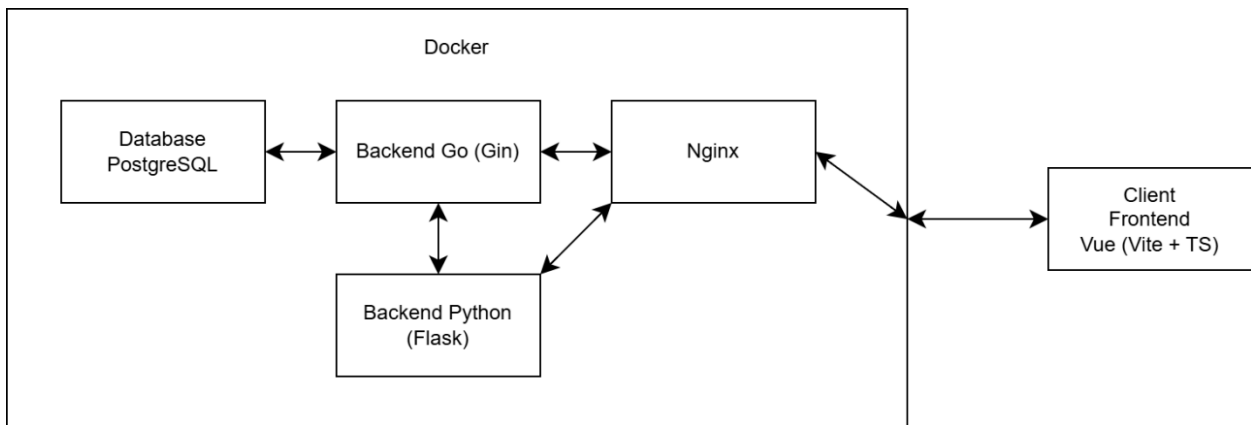


Рисунок 19 – Архитектура информационной системы

## 2.7.2 Спецификация компонентов

База данных:

- СУБД: PostgreSQL 16
- Назначение: Хранение структурированных данных, обработка SQL-запросов, выполнение транзакций.

Backend:

Основной:

- Язык: Go 1.24.0
- Фреймворк: Gin
- Библиотеки: pgx (драйвер PostgreSQL)
- Назначение: Основной API сервер, взаимодействие с БД, предоставление REST-интерфейсов.

Вспомогательный:

- Язык: Python 3.13.2
- Фреймворк: Flask
- Назначение: Генерация PDF-отчетов, обработка специализированных запросов к Go-серверу.

Frontend:

- Технологии: HTML, CSS, TypeScript
- Фреймворк: Vue 3
- Сборщик: Vite
- Назначение: Веб-интерфейс пользователя, отправка запросов к backend.

Реверс-прокси сервер:

- Сервер: Nginx
- Назначение: Роутинг запросов от клиента к фронтенду и backend-сервисам.

Система оркестрации и контейнеризации:

- Docker



### **2.7.3 Распределение бизнес-логики между компонентами**

Распределение бизнес логики между компонентами:

1. СУБД – Хранение данных, выполнение SQL-запросов, управление транзакциями, поддержка хранимых процедур и индексов.
2. Backend
  - a. Go (Gin):
    - i. Обработка HTTP-запросов от клиента.
    - ii. Валидация и обработка входных данных.
    - iii. Взаимодействие с базой данных через pgx.
    - iv. Формирование и возврат JSON-ответов.
  - b. Python (Flask):
    - i. Получение данных от Go-сервера.
    - ii. Формирование и генерация отчетов в формате PDF.
3. Frontend (Vue + Vite) – Отображение информации пользователю, отправка HTTP-запросов (через fetch/axios) к backend через Nginx, реактивный UI, обеспечивающий взаимодействие с системой.

#### **2.7.4 Интерфейсы взаимодействия компонентов**

Frontend и Nginx:

- Протокол: HTTP
- Назначение: Передача пользовательских запросов на backend или получение статических файлов.

Nginx и Backend (Go и Flask):

- Протокол: HTTP (REST API)
- Nginx маршрутизирует запросы в зависимости от URL к Go или Flask.

Backend (Go) и PostgreSQL:

- Протокол: PostgreSQL binary protocol
- Через драйвер pgx.

Flask и Go:

- Протокол: HTTP (внутренние запросы REST API)
- Flask получает данные через API Go-сервера для последующей генерации отчетов.

## 2.8 Хранимые процедуры и триггеры

### 2.8.1 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 20 представлена блок-схема алгоритма процедуры.



Рисунок 20 – 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 21 представлено содержимое таблицы scholarship до выполнения хранимой процедуры.

	id_scholarship [PK] integer	id_num_student integer	name_semester character varying (5)	size_scholarship money	id_budget integer
1	1	1220060404	2023O	5 000,00 ?	1
2	2	1220060502	2023O	5 200,00 ?	1
3	3	1220060402	2023O	5 400,00 ?	1
4	4	1220060401	2023O	5 700,00 ?	1
5	5	1220060402	2023O	15 500,00 ?	2
6	7	1220060404	2023O	15 100,00 ?	2
7	8	1220060405	2023O	5 400,00 ?	1
8	10	1220060407	2023O	5 300,00 ?	1
9	11	1220060503	2023B	5 700,00 ?	1
10	12	1220060402	2023B	5 300,00 ?	1
11	13	1220060401	2023B	5 400,00 ?	1
12	14	1220060401	2023B	15 500,00 ?	2
13	15	1220060402	2023B	15 600,00 ?	2
14	17	1220060404	2023B	5 200,00 ?	1
15	18	1220060405	2023B	5 500,00 ?	1
16	20	1220060407	2023B	5 300,00 ?	1

Рисунок 21 – Таблица scholarship до вызова процедуры

На рисунке 22 представлено содержимое таблицы scholarship после выполнения хранимой процедуры.

	id_scholarship [PK] integer	id_num_student integer	name_semester character varying (5)	size_scholarship money	id_budget integer
1	11	1220060503	2023B	5 700,00 ?	1
2	12	1220060402	2023B	5 300,00 ?	1
3	13	1220060401	2023B	5 400,00 ?	1
4	14	1220060401	2023B	15 500,00 ?	2
5	15	1220060402	2023B	15 600,00 ?	2
6	17	1220060404	2023B	5 200,00 ?	1
7	18	1220060405	2023B	5 500,00 ?	1
8	20	1220060407	2023B	5 300,00 ?	1
9	21	1220060402	2023O	4 800,00 ?	2
10	22	1220060407	2023O	4 650,00 ?	2

Рисунок 22 – Таблица scholarship после вызова процедуры

### 2.8.2 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 23 представлена таблица после выполнения запроса.

```
ERROR:  Оценка должна быть в диапазоне от 0 до 100. Некорректное значение: 150  
CONTEXT:  функция PL/pgSQL check_mark_score(), строка 4, оператор RAISE
```

```
ОШИБКА:  Оценка должна быть в диапазоне от 0 до 100. Некорректное значение: 150  
SQL state: P0001
```

Рисунок 23 – Пример выполнения триггера



### 2.8.3 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 24 представлена таблица после выполнения запроса.

18	26	1220060401	2023O	Математика	80	[null]
19	28	1220060405	2023B	Физика	82	Хорошо

Рисунок 24 – Пример выполнения триггера

### **3 Рабочий проект**

#### **3.1 SQL-скрипт создания структуры БД**

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

#### **3.2 SQL-скрипт триггеров и хранимых процедур**

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

#### **3.3 Текст программы**

Текст программы представлен по ссылке на моём github:  
<https://github.com/LiveisFpv/Go-web>

#### **3.4 Руководство пользователя**

Руководство пользователя, содержащее основные разделы представлены в приложении В.

## **Заключение**

В рамках выполнения курсового проекта была разработана информационная система «Стипендиатус», предназначенная для автоматизации процесса начисления и распределения стипендий в образовательном учреждении.

Проект включал в себя все этапы разработки: от анализа предметной области и проектирования архитектуры системы до реализации функционала и составления эксплуатационной документации. Были определены ключевые бизнес-процессы, построены модели данных, разработаны интерфейсы взаимодействия между компонентами и реализована серверная логика, включающая хранимые процедуры и триггеры для контроля целостности данных.

Система обладает следующими преимуществами:

- Упрощение взаимодействия между студентами, бухгалтерией и деканатом;
- Повышение точности и прозрачности расчета стипендий;
- Автоматизация рутинных операций и снижение нагрузки на сотрудников;
- Поддержка отчетности с возможностью выгрузки в удобные форматы.

В результате реализации проекта была достигнута основная цель – создание удобного, функционального и надежного веб-приложения, которое может быть использовано в образовательных организациях для повышения эффективности стипендиального учета.

Полученный результат подтвердил актуальность и практическую значимость автоматизации в управлении академическими процессами, а также продемонстрировал возможности применения современных технологий для решения реальных задач.

## 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 'Cemectp'::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**

#### **1. Область применения**

Настоящее руководство пользователя описывает назначение, функциональные возможности, особенности использования и эксплуатации информационной системы «Стипендиатус». Система разработана для автоматизации процессов начисления и распределения стипендий студентам образовательного учреждения.

#### **2. Краткое описание возможностей**

Система предназначена для трех категорий пользователей:

- Студентов, которые могут просматривать успеваемость, добавлять достижения и получать информацию о начислениях;
- Сотрудников деканата, выполняющих внесение данных и корректировку информации;
- Бухгалтерии, которая отвечает за распределение бюджета и генерацию отчетов.

#### **3. Уровень подготовки пользователя**

Уровень подготовки пользователя — базовое владение компьютером и браузером.

#### **4. Перечень эксплуатационной документации с которыми необходимо ознакомиться пользователю**

Перед началом работы рекомендуется ознакомиться с:

- Настоящим руководством;
- Пользовательским интерфейсом;
- Инструкциями администратора;
- Страницей о проекте на сайте

## **2 Назначения и условия применения**

1. Виды деятельности, функции, для автоматизации которых предназначено данное средство автоматизации.

Информационная система «Стипендиатус» предназначена для:

- Ведения базы студентов и их характеристик;
- Учета академической успеваемости;
- Автоматического расчета стипендий;
- Генерации отчетов по группам, успеваемости, бюджетам;
- Обеспечения прозрачного распределения стипендий на основе балльной системы и достижений студентов.

2. Условия, при соблюдении которых обеспечивается применение средства автоматизации.

- Наличие рабочей станции (ПК, ноутбук, планшет);
- Современный браузер (Google Chrome, Mozilla Firefox, Microsoft Edge);
- Подключение к внутренней сети учебного заведения или Интернету;
- Наличие учётной записи в системе.



### 3 Подготовка к работе

#### 1. Состав и содержание дистрибутивного носителя данных.

Состав системы:

- Веб-клиент (интерфейс пользователя);
- Серверная часть (Go GIN);
- Сервис отчетности (Python Flask);
- База данных (PostgreSQL);
- Инфраструктура контейнеров (Docker, Nginx).


Разработанное программное обеспечение поставляется в виде исходного кода, для запуска на локальном и стороннем сервере через Docker и дальнейшем доступом через любой браузер.

#### 2. Порядок загрузки данных и программ.

- Скопируйте исходный код в папку на сервере.
- Пропишите обязательные параметры в env файле
- Запустите через Docker compose

#### 3. Порядок проверки работоспособности.

После успешного запуска web-приложения в нем будет необходимо зарегистрироваться. После успешной авторизации клиент попадёт на главную и могут использовать доступный функции системы их роли пользователя. На рисунках 25-26 представлена форма авторизации и главная страница.



### Авторизация


Email

Некорректный формат email

Пароль

Войти


или


 Войти через Google

[Создать аккаунт](#)



Рисунок 25 – Страница авторизации




Профиль 

## Стипендиатус


Эффективное управление образовательным процессом

### Основные модули системы




Студенты

Управление данными студентов




Группы

Управление учебными группами




Оценки


Управление успеваемостью



Календарь



Финансы



Документы




Рисунок 26 – Главная страница

#### 4 Описание операций

В таблице 7 представлено описание операций, производимых в системе.

Таблица 7 – Описание операций

Функция	Задача	Описание
Вход в систему	Авторизация пользователя	Пользователь вводит логин и пароль и получает соответствующую роль в системе
Просмотр профиля	Ознакомление с личной информацией	В разделе "Профиль" отображаются ФИО, номер студенческого билета, группа, email. Возможность изменения некорректных данных.
Добавление достижения	Подача на повышенную стипендию	Студент заполняет форму достижения, указывает название, категорию и дату получения. Информация проходит модерацию деканатом.
Просмотр оценок	Ознакомление с успеваемостью	Студент видит оценки по дисциплинам, выставленные деканатом. Отображается тип оценки и семестр.
Редактирование студентов	Управление списком студентов	Деканат добавляет, редактирует или удаляет записи о студентах. Вводятся ФИО, номер билета, email, группа.
Внесение оценок	Заполнение академических результатов	Деканат выставляет оценки студентам, указывая дисциплину, баллы, семестр. При необходимости правит оценки.
Редактирование достижений	Управление системой достижений	Деканат подтверждает или отклоняет достижения студентов. Также возможна правка некорректных данных.
Просмотр бюджета	Контроль доступных средств	Бухгалтерия отслеживает выделенные суммы на стипендии в конкретный семестр. Доступен просмотр бюджета по категориям.

Назначение стипендий	Автоматический расчет выплат	Выполняется через запуск процедуры. На основе баллов и бюджета формируются выплаты.
Просмотр назначений	Ознакомление с результатами распределения	Отображается список студентов, получивших стипендию, с указанием суммы и типа.
Формирование отчетов	Генерация документации	Возможность загрузки отчетов по успеваемости, достижениям, начислениям в PDF.
Фильтрация данных	Поиск нужной информации	Таблицы поддерживают фильтрацию по группе, фамилии, семестру, типу стипендии и другим параметрам.
Использование модальных окон	Удобное редактирование записей	При изменении записей открывается модальное окно без выхода со страницы. Это упрощает ввод и предотвращает потерю контекста.
Отбор по периоду	Анализ по временным рамкам	При формировании отчетов сохраняется фильтрация для получения точных данных.
Выход из системы	Завершение работы	После завершения работы пользователь нажимает кнопку "Выйти" и сеанс завершается.

## 5 Аварийные ситуации

В таблице 8 представлено описание ошибок и их возможное решение.

Таблица 8 – Описание ошибок

Класс ошибки	Ошибка	Описание ошибки	Действия пользователя в случае возникновения данных ошибок
Ошибка авторизации	Указан неправильный логин или пароль	Пользователь неверно указал пару логин и пароль	Проверить корректность введенных данных, раскладку клавиатуры и случайное нажатие клавиш
Ошибки валидации	Введены неправильные данные в модальной форме	Пользователь ввел не валидные данные	Проверить корректность введенных данных

## **6 Рекомендации к освоению**

Для успешного освоения разработанной системы необходимы базовые знания пользователя компьютера и навыки работы с веб-приложениями. Важно, чтобы пользователи были знакомы с основными функциями системы и могли эффективно использовать ее возможности для выполнения повседневных задач. Начните работу с изучения интерфейса после входа. Используйте меню слева для перехода между функциями. При редактировании данных соблюдайте формат. Добавляйте достижения заранее, чтобы они учитывались при расчете. При необходимости — используйте фильтры для поиска информации в меню справа сверху над таблицей. Обращайтесь к администратору при подозрении на сбой или ошибку. Регулярно обновляйте браузер для корректной работы интерфейса. Соблюдайте политику безопасности при работе с личными данными.