Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ (ИУ)
КАФЕДРА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)

КУРСОВАЯ РАБОТА

НА ТЕМУ:

Разработка базы данных электронного дневника

Студент	ИУ7-63Б		Лазутин А.В.
	(группа)	(подпись, дата)	(И.О. Фамилия)
Руководитель курсового проекта			Шибанова Д.А.
		(подпись, дата)	(И.О. Фамилия)

2024 г.

РЕФЕРАТ

Расчетно-пояснительная записка 53 с., 11 рис., 16 табл., 25 ист.

БАЗЫ ДАННЫХ, ЭЛЕКТРОННЫЙ ДНЕВНИК, PostgreSQL, KOTLIN, РОЛИ БАЗЫ ДАННЫХ, ТРИГГЕР.

Цель работы – разработка базы данных для электронного дневника.

Результатом работы являются разработанная база данных и приложение для взаимодействия с базой данных. Приложение предоставляет возможности просмотра, изменения, добавления и удаления данных в зависимости от роли пользователя. В качестве системы управления базами данных используется PostgreSQL.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Электронный дневник — приложение, с помощью которого дети и родители могут посмотреть оценки, узнать расписание уроков и взять книгу из библиотеки. Электронный дневник не только позволяет экономить время как ученикам, так и учителям, а также дает родителям возможность следить за успеваемостью ребёнока в школе. Преподаватели могут выставлять в дневнике оценки и смотреть их у своих учеников. Целью данной работы является разработка базы данных для электронного дневника.

Чтобы достигнуть поставленной цели, необходимо решить следующие задачи:

— провести анализ предметной области электронного дневника;

— спроектировать базу данных, описать сущности, проектируемую ролевую модель базы данных;

— разработать программную реализацию приложения, описать интерфейс доступа к базе данных;

— исследовать зависимость времени выполнения запроса от количества записей в таблице.

1 Аналитическая часть

В этом разделе будет проведён анализ предментой области и обзор существующих решений, обоснована актуальность решаемой продуктом задачи. Также, будет сформировано описание пользователей проектируемого приложения электронного дневника для доступа к базе данных и выбрана модель данных.

1.1 Сравнение существующих решений

Одной из наиболее актуальных технологий, которые активно вводятся в образовании, на сегодняшний день является система электронных дневников [1]. На данный момент в России существуют несколько крупных информационных систем, которые помогут автоматизировать образовательный процесс и убрать часть бумажной работы: NetSchool [2], SmileS.Школьная Карта [3], Аверс [4].

В таблице 1.1 представлен сравнительный анализ известных решений [5].

Таблица 1.1 – Сравнение известных решений

Название	Личный кабинет	Бесплатное использование	Библиотека
NetSchool	+	-	-
SmileS.Школьная Карта	+	+	-
Аверс	-	-	-

Таким образом, разработка нового школьного дневника с расширенными функциональными возможностями, а также без каких-либо ограничений на использование, стала необходимостью. Это позволит создать более удобную и эффективную среду для взаимодействия всех участников образовательного процесса, повысить качество образования и обеспечить доступ к современным образовательным ресурсам.

1.2 Формализация задачи

В рамках курсовой работы необходимо спроектировать и разработать базу данных для электронного дневника, а также приложение, позволяющее с ней работать. Разрабатываемое решение должно удовлетворять всем критериям: у каждого пользователя должен быть личный кабинет, приложение должно быть в открытом доступе, и оно должно позволять ученикам пользоваться библиотекой. В электронном дневнике имеется разграничение по ролям: гость, ученик, учитель и администратор. Подробное описание их возможностей представлено на рисунке 1.1.

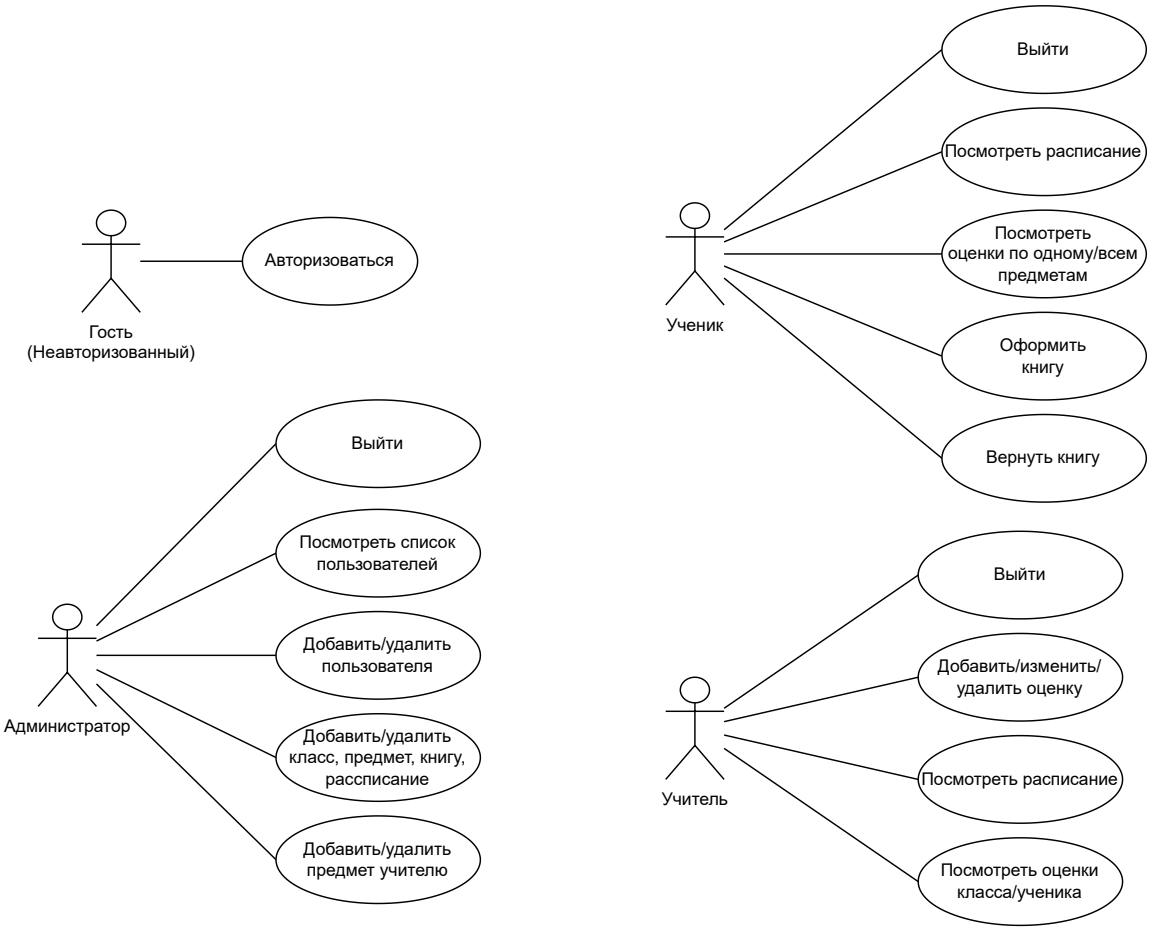

Рисунок 1.1 – Диаграмма использования приложения

1.3 Формализация данных

Разрабатываемая база данных должна хранить информацию о:

— Класс;

— Ученик;

— Учитель;

— Рассписание;

— Оценки;

— Библиотека;

— Предмет.

На рисунке 1.2 представлена ER-диаграмма сущностей в нотации Чена, описывающая сущности, их атрибуты и связи между ними.

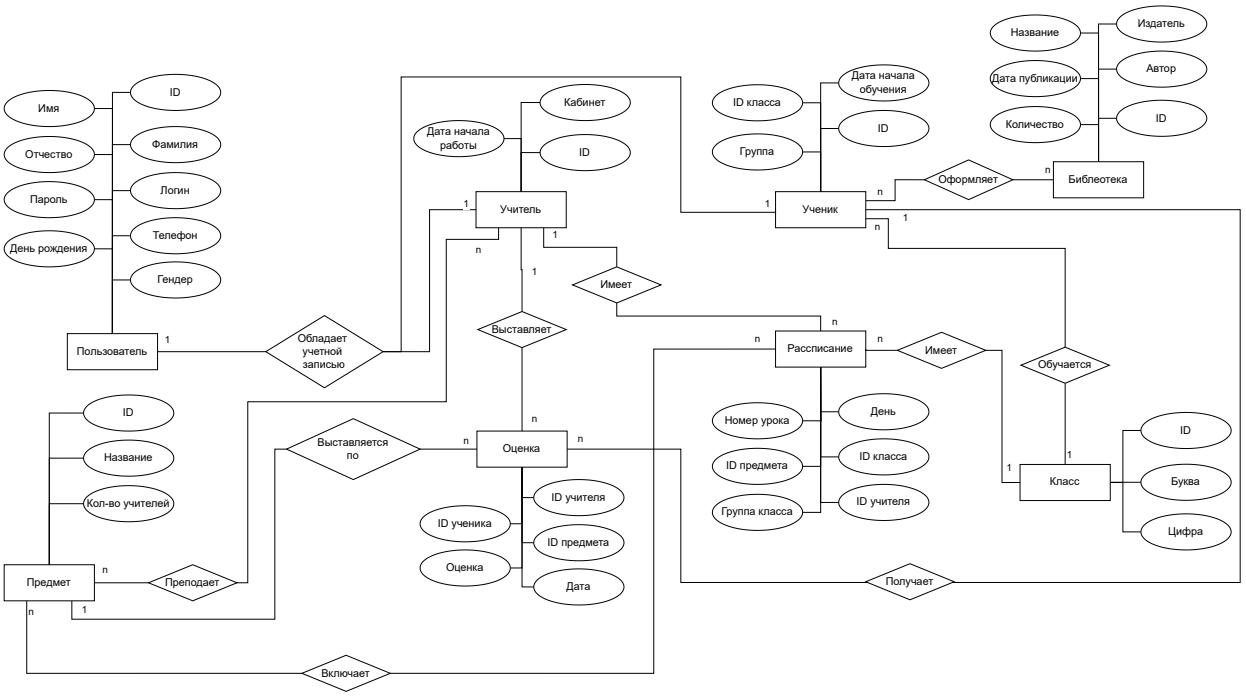

Рисунок 1.2 – ER-диаграмма

1.4 Анализ баз данных

Модель данных – это совокупность структур данных и операций их обработки. С помощью модели данных могут быть представлены информа-

ционные объекты и взаимосвязи между ними. По модели хранения базы данных делятся на три группы [6]:

— дореляционная;

— реляционная;

— постреляционная.

Различные модели хранения данных обеспечивают разные механизмы для работы с информацией, что позволяет выбирать наиболее подходящую модель для конкретной задачи или приложения

1.4.1 Дореляционная база данных

К дореляционным базам данных относятся:

— инвертированные списки;

— иерархические;

— сетевые.

База данных, основанная на инвертированных списках, состоит из множества файлов, в которых записи упорядочены согласно физической организации данных. Для каждого такого файла можно определить любое количество дополнительных порядков на основе инвертированных списков.

Иерархическая модель хранения данных использует древовидную структуру, где данные организованы в виде объектов с указателями от родительских элементов к дочерним.

Основные концепции сетевой модели включают элементы (узлы) и связи. Узел представляет собой набор атрибутов данных, описывающих определенный объект. Сетевая модель не является полностью независимой от приложений, так как извлечение данных определяется физической структурой хранения. Следовательно, изменение структуры данных требует внесения изменений в приложение [7].

1.4.2 Реляционная база данных

Реляционные базы данных чаще всего используют язык SQL [8]. Данные в таких базах хранятся в виде таблиц и строк, а таблицы могут быть связаны с другими таблицами через внешние ключи, создавая отношения [9].

Структура реляционных баз данных позволяет связывать информацию из разных таблиц посредством внешних ключей, которые обеспечивают уникальную идентификацию любого атомарного фрагмента данных в таблице. Другие таблицы могут ссылаться на эти внешние ключи для установления связи между частями данных.

Важным свойством реляционных баз данных является их способность удовлетворять требованиям ACID: атомарность, согласованность, изоляция и устойчивость [10].

1.4.3 Постреляционная база данных

Постреляционная модель данных представляет собой расширенную реляционную модель, которая снимает ограничение на неделимость данных в записях таблиц. Такие базы данных поддерживают работу со сложными типами данных, создаваемыми пользователями. Однако эти решения не всегда могут обеспечить полную поддержку ACID.

Нереляционные хранилища данных могут использоваться совместно с реляционными базами данных. Например, в системах, где основная информация хранится в SQL-базах данных, а за кэширование отвечает нереляционная база данных [11].

1.4.4 Выбор модели данных

Рассмотрим несколько факторов, исходя из которых будет выбрана модель данных:

— выделенные в текущем разделе семь сущностей представляют собой структурированные данные, структура которых не подвержена частым

изменениям;

— важно соблюдать целостность и структуризацию хранимых данных;

— выбираемая модель данных должна соответствовать требованиям к транзакционным системам.

— в рамках проектируемой базы данных отсутствует необходимость хранения коллекций.

Исходя из перечисленных причин, подходящей моделью данных является реляционная модель данных.

Вывод

В этом разделе был проведён анализ предментой области и обзор существующих решений, на его основе формализована задача и описаны требования к пользовательскому интерфейсу для взаимодействия с базой данных. Для обеспечения безопасной работы с данными разработана политика управления доступом к данным. Проведена формализация данных и классификация моделей данных.

С учетом особенности задачи была выбрана реляционная модель хранения данных, так как предметная область может быть представлена в виде «таблиц» и должна удовлетворять требованиям ACID.

2 Конструкторская часть

В данном разделе будет проведена формализация сущностей проектируемой системы. Описываются ограничения целостности электронного дневника и ролевая модель.

2.1 Формализация сущностей системы

Обозначения в таблицах:

— PK — первичный ключ;

— FK — внешний ключ;

— U — уникальное значение;

— NN — поле не может принимать неопределённое значение;

— I — целочисленный тип;

— DT — тип временной отметки;

— S — символьный тип;

— ID — тип идентификатора.

Также для корректного представления информации в базе данных, были введи дополнительные типы:

— StudentAssessmentType (SAT) — перечисляемый тип данных для проверки принадлежности оценки одному из значений: «ONE», «TWO», «THREE», «FOUR», «FIVE»;

— ClassNumberType (CNT) — перечисляемый тип данных для проверки принадлежности номеру класса одному из значений: «ONE», «TWO», «THREE», ..., «TEN», «ELEVEN»;

— GroupScheduleType (GScT) — перечисляемый тип данных для проверки принадлежности урока в рассписании одному из значений: «FIRST», «SECOND», «JOINT»;

— GroupStudentType (GStT) — перечисляемый тип данных для проверки принадлежности группы ученика одному из значений: «FIRST», «SECOND»;

— GenderType (GT) — перечисляемый тип данных для проверки принадлежности пола пользователя одному из значений: «MALE», «FEMALE»;

— CountGroupType (CGT) — перечисляемый тип данных для проверки принадлежности количество групп у предмета одному из значений: «ONE», «TWO»;

— DayOfWeekType (DWT) — перечисляемый тип данных для проверки принадлежности дня недели одному из значений: «MONDAY», «TUESDAY», ..., «FRIDAY», «SATURDAY»;

— NameRoleType (NRT) — перечисляемый тип данных для проверки принадлежности роли одному из значений: «STUDENT», «TEACHER», «ADMIN»;

2.1.1 Таблица Class

В таблице Class содержится информация о классах школы. Описание полей таблицы Class представлено в таблице 2.1.

Таблица 2.1 – Описание полей таблицы Client

Название	Характеристики					
	PK	FK	U	NN	Тип	Значение
classID	+	-	+	+	ID	Идентификатор класса
classLetter	-	-	-	+	S	Буква класса
classNumber	-	-	-	+	CNT	Номер класса

2.1.2 Таблица User

В таблице User содержится информация о пользователях школы. Описание полей таблицы Student представлено в таблице 2.2.

Таблица 2.2 – Описание полей таблицы User

Название	Характеристики					
	PK	FK	U	NN	Тип	Значение
UserID	+	-	+	+	ID	Идентификатор
FirstName	-	-	-	+	S	Имя пользователя
LastName	-	-	-	+	S	Фамилия пользователя
Patronymic	-	-	-	+	S	Отчество пользователя
Birthday	-	-	-	+	DT	Дата рождения
Login	-	-	+	+	S	Логин
Password	-	-	-	+	S	Пароль
PhoneNumber	-	-	-	+	S	Номер телефона
Gender	-	-	-	+	GT	Пол пользователя

2.1.3 Таблица Student

В таблице Student содержится информация о учениках школы. Описание полей таблицы Student представлено в таблице 2.3.

Таблица 2.3 – Описание полей таблицы Student

Название	Характеристики					
	PK	FK	U	NN	Тип	Значение
StudentID	-	+	+	+	ID	Идентификатор ученика
ClassID	-	+	-	+	ID	Идентификатор класса
StartStudy	-	-	-	+	DT	Дата начала обучения
Group	-	-	-	+	GStT	Группа ученика

2.1.4 Таблица Teacher

В таблице Teacher содержится информация о учителях школы. Описание полей таблицы Teacher представлено в таблице 2.4.

Таблица 2.4 – Описание полей таблицы Teacher

Название	Характеристики					
	PK	FK	U	NN	Тип	Значение
TeacherID	-	+	+	+	ID	Идентификатор учителя
StartTeach	-	-	-	+	DT	Дата начала преподавания
Cabinet	-	-	+	+	I	Кабинет учителя

2.1.5 Таблица Subject

В таблице Subject содержится информация о преподоваемых предметах. Описание полей таблицы Subject представлено в таблице 2.5.

Таблица 2.5 – Описание полей таблицы Subject

Название	Характеристики					
	PK	FK	U	NN	Тип	Значение
SubjectID	+	-	+	+	ID	Идентификатор
NameSubject	-	-	+	+	S	Название предмета
CountTeachers	-	-	-	+	CGT	Количество подгрупп

2.1.6 Таблица Schedule

В таблице Schedule содержится информация о рассписании. Описание полей таблицы Schedule представлено в таблице 2.6.

Таблица 2.6 – Описание полей таблицы Schedule

Название	Характеристики					
	PK	FK	U	NN	Тип	Значение
Day	-	-	-	+	DWT	День недели
GroupClass	-	-	-	+	GScT	Группа
ClassID	-	+	-	+	ID	Идентификатор класса
SubjectID	-	+	-	+	ID	Идентификатор предмета
TeacherID	-	+	-	+	ID	Идентификатор учителя
LessinNumber	-	-	-	+	I	Номер урока

2.1.7 Таблица Assessment

В таблице Assessment содержится информация о полученных оценках ученика. Описание полей таблицы Assessment представлено в таблице 2.7.

Таблица 2.7 – Описание полей таблицы Assessment

Название	Характеристики					
	PK	FK	U	NN	Тип	Значение
StudentID	-	+	-	+	ID	Идентификатор ученика
SubjectID	-	+	-	+	ID	Идентификатор предмета
TeacherID	-	+	-	+	ID	Идентификатор учителя
Assessment	-	-	-	+	SAT	Полученная оценка
Date	-	-	-	+	DT	Дата получения оценки

2.1.8 Таблица Library

В таблице Library содержится информация о книгах. Описание полей таблицы Library представлено в таблице 2.8.

Таблица 2.8 – Описание полей таблицы Library

Название	Характеристики					
	PK	FK	U	NN	Тип	Значение
BookID	+	-	+	+	ID	Идентификатор книги
Title	-	-	+	+	S	Название книги
Author	-	-	-	+	S	Автор книги
DatePublication	-	-	-	+	DT	Дата публикации
Publisher	-	-	-	+	S	Издатель
Count	-	-	-	+	I	Кол-во книг

2.1.9 Таблица Role

В таблице Role содержится информация о ролях. Описание полей таблицы Role представлено в таблице 2.9.

Таблица 2.9 – Описание полей таблицы Role

Название	Характеристики					
	PK	FK	U	NN	Тип	Значение
RoleID	+	-	+	+	ID	Идентификатор роли
NameRole	-	-	+	+	NRT	Название роли

2.1.10 Таблица UserRole

В таблице UserRole содержится информация о пользователе и его роли. Описание полей таблицы UserRole представлено в таблице 2.10.

Таблица 2.10 – Описание полей таблицы UserRole

Название	Характеристики					
	PK	FK	U	NN	Тип	Значение
UserID	-	+	+	+	ID	Идентификатор пользователя
RoleID	-	+	-	+	S	Идентификатор роли

2.1.11 Таблица StudentBook

В таблице StudentBook содержится информация об ученика и его оформленных книгах. Описание полей таблицы StudentBook представлено в таблице 2.11.

Таблица 2.11 – Описание полей таблицы StudentBook

Название	Характеристики					
	PK	FK	U	NN	Тип	Значение
BookID	-	+	-	+	ID	Идентификатор книги
StudentID	-	+	-	+	ID	Идентификатор ученика

2.1.12 Таблица TeacherSubject

В таблице TeacherSubject содержится информация об учителе и предметах, которые он преподает. Описание полей таблицы TeacherSubject представлено в таблице 2.12.

Таблица 2.12 – Описание полей таблицы TeacherSubject

Название	Характеристики					
	PK	FK	U	NN	Тип	Значение
TeacherID	-	+	-	+	ID	Идентификатор учителя
SubjectID	-	+	-	+	S	Идентификатор предмета

2.2 Ограничения

Для контроля валидности данных введены следующие ограничения:

— check_phone_number — символьный тип данных для проверки номера телефона с помощью регулярного выражения;

— check_before_current — тип временной отметки данных с ограничением не позже, чем текущее время;

— check_positive_number — целочисленный тип данных с ограничением не меньше 0;

На рисунке 2.1 представлена ER-диаграмма сущностей в нотации Мартина.

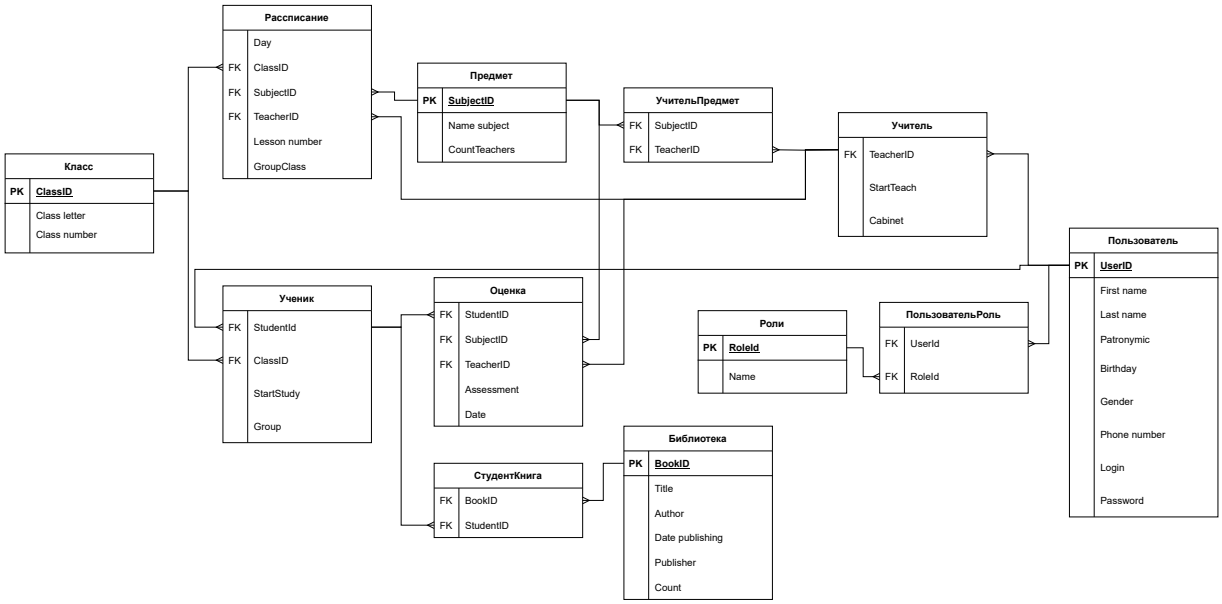

Рисунок 2.1 – ER-диаграмма сущностей в нотации Мартина

2.3 Ролевая модель на уровне базы данных

На уровне взаимодействия с базой данных представлена следующая ролевая модель:

1) Гость должен обладать правами на:

— SELECT в таблицы User, UserRole, Student, Teacher;

2) Ученик должен обладать правами на:

— SELECT в таблицы Class, User, UserRole, Student, Teacher, Subject, Schedule, Assessment, Library, StudentBook;

— INSERT, DELETE в таблицу StudentBook;

— UPDATE в таблицу Library;

3) Учитель должен обладать правами на:

— SELECT в таблицы Class, User, UserRole, Teacher, Student, Subject, Schedule, Assessment, TeacherSubject;

— INSERT, DELETE, UPDATE в таблицу Assessment;

4) Администратор должен иметь права на все действия со всеми таблицами.

2.4 Триггер базы данных

При добавления новой книги в библеотеку нужно убедиться, что информация о ней корректна. Для этого необходим триггер, реализующий проверку валидности добавляемой книги. При попытке добавления новой записи в таблицу Library вызывается функция, схема алгоритма работы которой представлена на рисунке 2.2.

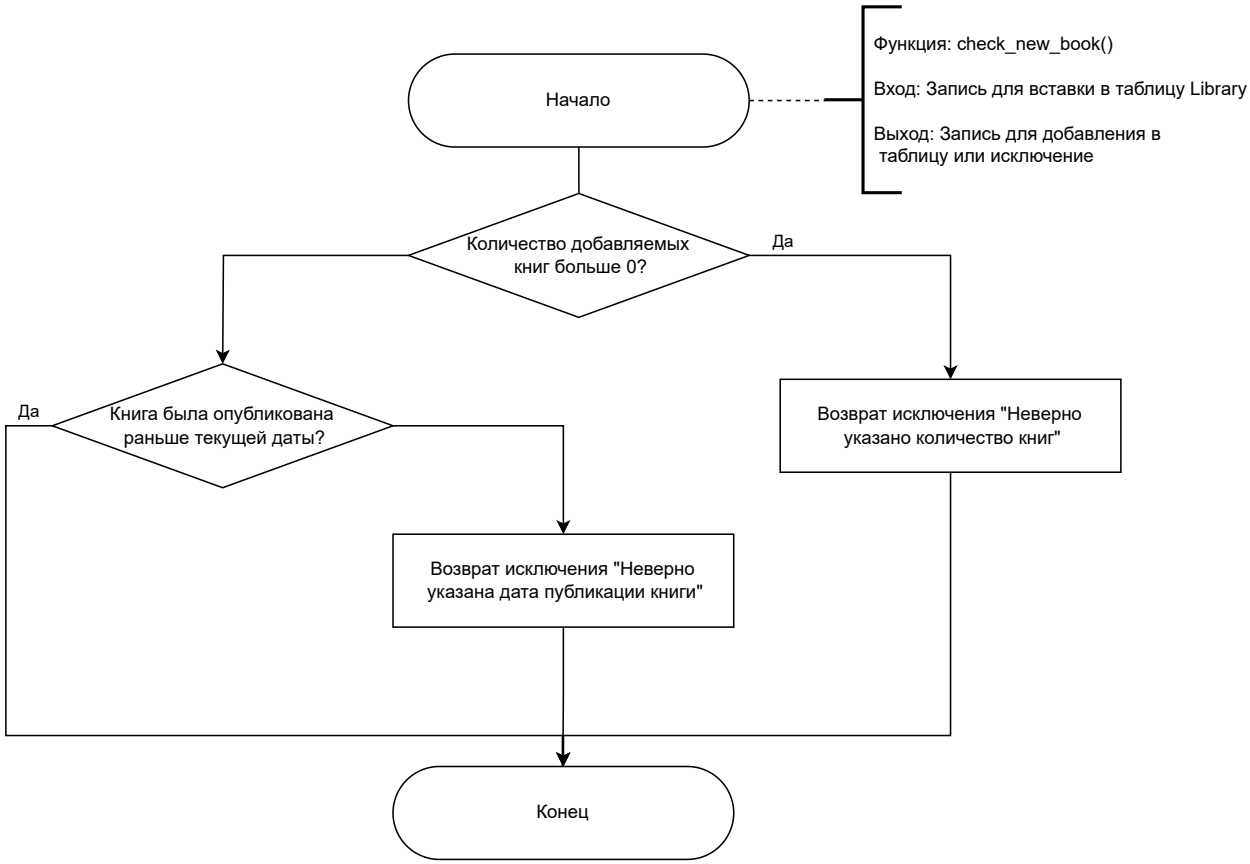

Рисунок 2.2 – Схема алгоритма функции проверки новой записи check_new_book()

Вывод

В данном разделе были выделеные сущности: класс, ученик, учитель, рассписание, оценки, библеотека, предмет. Была спроектирована база данных: на основе ER-диаграммы сущностей описаны таблицы и отношения между этими таблицами в форме диаграммы базы данных в нотации Мартина. Также описан проектируемый триггер базы данных и роли базы данных: гость, ученик, учитель и администратор.

3 Технологическая часть

В данной части рассматривается выбор средств реализации. Описываются методы тестирования разработанного функционала и приводится интерфейс программного обеспечения.

3.1 Выбор системы управления базами данных

Существует большое количество реляционных систем управления базами данных (СУБД), и одни из популярных это [12] — Oracle [13], MySQL [14], PostgreSQL [15].

Для сравнения выбранных СУБД выделены следующие критерии:

— бесплатное программное обеспечение;

— производительность [16];

— поддержка ACID-транзакций.

Таблица 3.1 – Сравнение СУБД

СУБД	Бесплатное программное обеспечение	Производительность	Поддержка ACID-транзакций
Oracle	-	2	+
PostgreSQL	+	1	+
MySQL	+	3	+

По результатам сравнения, полученных в таблице 3.1, в качестве СУБД для реляционной модели баз данных выбран PostgreSQL.

3.2 Средства реализации программного обеспечения

В качестве языка программирования был выбран Kotlin [17]. Этот выбор объясняется тем, что Kotlin является объектно-ориентированным языком программирования, что позволяет создавать классы, объекты и методы, облегчая взаимодействие с объектами. Также язык имеет строгую типизацию данных, в нем имеется множество библеотек для тестирования как целого ПО, так и его некоторой части. И он обаладает средствами для шифрования данных и взаимодествия с PostgreSQL.

В качестве среды разработки было выбрано IntelliJ IDEA [18]. Данная среда разработки обеспечивает высокий уровень поддержки и интеграции с языком программирования Kotlin. Она предлагает мощные инструменты для рефакторинга, статического анализа и навигации, а также удобные средства для отладки и профилирования. Интеграция с инструментами сборки и поддержка различных фреймворков для тестирования делают процесс разработки более эффективным и управляемым, имеет множество встроенных плагинов. Все это необходимо для успешной разработки ПО.

Также для безопасности хранимых данных были предусмотрены определённые средства защиты:

— аутентификация пользователя производится по его логину и паролю. Для повышения безопасности хранения паролей в базе данных применяется шифрование пароля с помощью хэш-функции SHA-256 [19];

— после успешной аутентификации пользователю назначается роль в соответствии с ролевой моделью, которая ограничивает доступ отдельных пользователей к данным хранилища;

— каждый запрос в базу данных использует для защиты от SQL-инъекции PreparedStatement [20]. Это гарантирует, что введенные пользователем данные не могут изменить структуру исходного SQL-запроса.

3.3 Создание таблиц

В листингах 3.1 — 3.12 представлено создание таблиц базы данных.

Листинг 3.1 – Создание таблицы Class

```
CREATE TABLE IF NOT EXISTS diary.Class(
    ClassId SERIAL,
    ClassLetter VARCHAR(1),
    ClassNumber ClassNumberType
);
```

Листинг 3.2 – Создание таблицы User

```
CREATE TABLE IF NOT EXISTS diary.User(
    UserId SERIAL,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    Patronymic VARCHAR(100),
    Birthday Date,
    Password VARCHAR(100),
    Identifier VARCHAR(100) UNIQUE,
    Phone VARCHAR(100),
    Gender GenderType
);
```

Листинг 3.3 – Создание таблицы Role

```
CREATE TABLE IF NOT EXISTS diary.Role(
    RoleId SERIAL,
    NameRole NameRole UNIQUE
);
```

Листинг 3.4 – Создание таблицы UserRole

```
CREATE TABLE IF NOT EXISTS diary.UserRole(
    UserId Int UNIQUE,
    RoleId Int
);
```

Листинг 3.5 – Создание таблицы Student

```
CREATE TABLE IF NOT EXISTS diary.Student(
    StudentId Int UNIQUE,
    DateStartStuding Date,
    ClassId Int,
    GroupStudent GroupStudentType
);
```

Листинг 3.6 – Создание таблицы Teacher

```
CREATE TABLE IF NOT EXISTS diary.Teacher(
    TeacherId Int UNIQUE,
    DateStartTeaching Date,
    Cabinet Int UNIQUE
);
```

Листинг 3.7 – Создание таблицы Subject

```
CREATE TABLE IF NOT EXISTS diary.Subject(
    SubjectId SERIAL,
    Name VARCHAR(100) UNIQUE,
    CountTeachers CountGroupType
);
```

Листинг 3.8 – Создание таблицы Schedule

```
CREATE TABLE IF NOT EXISTS diary.Schedule(
    DayOfWeek DayOfWeekType,
    LessonNumber Int,
    GroupStudent GroupScheduleType,
    ClassId Int,
    SubjectId Int,
    TeacherId Int
);
```

Листинг 3.9 – Создание таблицы Assessment

```
CREATE TABLE IF NOT EXISTS diary.Assessment(
    StudentId Int,
    SubjectId Int,
    TeacherId Int,
    Assessment StudentAssessmentType,
    Date Date
);
```

Листинг 3.10 – Создание таблицы Library

```
CREATE TABLE IF NOT EXISTS diary.Library(
    BookId SERIAL,
    Title VARCHAR(100) UNIQUE,
    Author VARCHAR(100),
    DatePublication Date,
    Publisher VARCHAR(100),
    Count Int
);
```

Листинг 3.11 – Создание таблицы StudentBook

```
CREATE TABLE IF NOT EXISTS diary.StudentBook(
    BookId Int,
    StudentID Int
);
```

Листинг 3.12 – Создание таблицы TeacherSubject

```
CREATE TABLE IF NOT EXISTS diary.TeacherSubject(
    TeacherID Int,
    SubjectId Int
);
```

3.4 Ограничения целостностей

В листингах 3.13 — 3.24 представлено создание ограничений целостности, наклабываемые на поля таблиц базы данных.

Листинг 3.13 – Создание ограничений целостности таблицы Class

```
ALTER table diary.Class
    ADD CONSTRAINT pk_class_id primary key(ClassId);
ALTER table diary.Class
    ALTER COLUMN ClassNumber SET NOT NULL,
    ALTER COLUMN ClassLetter SET NOT NULL;
```

Листинг 3.14 – Создание ограничений целостности таблицы User

```
ALTER table diary.User
    ADD CONSTRAINT pk_user_id primary key(UserId);
ALTER table diary.User
    ADD CONSTRAINT check_phone_number CHECK (Phone ~
        '^\+\d{1,3}\d{10}$'),
    ADD CONSTRAINT check_birthday_before_current CHECK (Birthday
       <= CURRENT_DATE),
    ALTER COLUMN FirstName SET NOT NULL,
    ALTER COLUMN LastName SET NOT NULL,
    ALTER COLUMN Patronymic SET NOT NULL,
    ALTER COLUMN Birthday SET NOT NULL,
    ALTER COLUMN Password SET NOT NULL,
    ALTER COLUMN Identifier SET NOT NULL,
    ALTER COLUMN Phone SET NOT NULL,
    ALTER COLUMN Gender SET NOT NULL;
```

Листинг 3.15 – Создание ограничений целостности таблицы Role

```
ALTER table diary.Role
    ADD CONSTRAINT pk_role_id primary key(RoleId);
ALTER table diary.Role
    ALTER COLUMN NameRole SET NOT NULL;
```

Листинг 3.16 – Создание ограничений целостности таблицы UserRole

```
ALTER table diary.UserRole
    ADD CONSTRAINT fk_userId foreign key(UserId) references
        diary.User(UserId) ON DELETE CASCADE,
    ADD CONSTRAINT fk_RoleId foreign key(RoleId) references
        diary.Role(RoleId) ON DELETE CASCADE,
    ALTER COLUMN UserId SET NOT NULL,
    ALTER COLUMN RoleId SET NOT NULL;
```

Листинг 3.17 – Создание ограничений целостности таблицы Student

```
ALTER table diary.Student
    ADD CONSTRAINT fk_classId foreign key(ClassId) references
        diary.Class(ClassId) ON DELETE CASCADE,
    ADD CONSTRAINT fk_studentId foreign key(StudentId)
        references diary.User(UserId) ON DELETE CASCADE,
    ADD CONSTRAINT check_date_before_current CHECK
        (DateStartStuding <= CURRENT_DATE),
```

```
5     ALTER COLUMN DateStartStuding SET NOT NULL,
6     ALTER COLUMN GroupStudent SET NOT NULL;
```

Листинг 3.18 – Создание ограничений целостности таблицы Teacher

```
1 ALTER table diary.Teacher
2     ADD CONSTRAINT fk_teacherId foreign key(TeacherId)
          references diary.User(UserId) ON DELETE CASCADE,
3     ADD CONSTRAINT check_date_before_current CHECK
          (DateStartTeaching <= CURRENT_DATE),
4     ALTER COLUMN DateStartTeaching SET NOT NULL,
5     ALTER COLUMN Cabinet SET NOT NULL;
```

Листинг 3.19 – Создание ограничений целостности таблицы Subject

```
1 ALTER table diary.Subject
2     ADD CONSTRAINT pk_subject_id primary key(SubjectId);
3 ALTER table diary.Subject
4     ALTER COLUMN Name SET NOT NULL,
5     ALTER COLUMN CountTeachers SET NOT NULL;
```

Листинг 3.20 – Создание ограничений целостности таблицы Schedule

```
1 ALTER table diary.Schedule
2     ADD CONSTRAINT fk_classId foreign key(ClassId) references
          diary.Class(ClassId) ON DELETE CASCADE,
3     ADD CONSTRAINT fk_subjectId foreign key(SubjectId)
          references diary.Subject(SubjectId) ON DELETE CASCADE,
4     ADD CONSTRAINT fk_teacherId foreign key(TeacherId)
          references diary.Teacher(TeacherId) ON DELETE CASCADE,
5     ALTER COLUMN LessonNumber SET NOT NULL,
6     ALTER COLUMN GroupStudent SET NOT NULL,
7     ALTER COLUMN ClassId SET NOT NULL,
8     ALTER COLUMN SubjectId SET NOT NULL,
9     ALTER COLUMN TeacherId SET NOT NULL,
10    ALTER COLUMN DayOfWeek SET NOT NULL;
```

Листинг 3.21 – Создание ограничений целостности таблицы Assessment

```
1 ALTER table diary.Assessment
2     ADD CONSTRAINT fk_studentId foreign key(StudentId)
          references diary.Student(StudentId) ON DELETE CASCADE,
```

```
3     ADD CONSTRAINT fk_subjectId foreign key(SubjectId)
          references diary.Subject(SubjectId) ON DELETE CASCADE,
4     ADD CONSTRAINT check_date_before_current CHECK (Date <=
          CURRENT_DATE),
5     ALTER COLUMN StudentId SET NOT NULL,
6     ALTER COLUMN SubjectId SET NOT NULL,
7     ALTER COLUMN TeacherId SET NOT NULL,
8     ALTER COLUMN Assessment SET NOT NULL,
9     ALTER COLUMN Date SET NOT NULL;
```

Листинг 3.22 – Создание ограничений целостности таблицы Library

```
1 ALTER table diary.Library
2     ADD CONSTRAINT pk_book_id primary key(BookId);
3 ALTER table diary.Library
4     ADD CONSTRAINT check_date_before_current CHECK
          (DatePublication <= CURRENT_DATE),
5     ADD CONSTRAINT check_positive_number CHECK (Count >= 0),
6     ALTER COLUMN BookId SET NOT NULL,
7     ALTER COLUMN Title SET NOT NULL,
8     ALTER COLUMN Author SET NOT NULL,
9     ALTER COLUMN DatePublication SET NOT NULL,
10    ALTER COLUMN Publisher SET NOT NULL,
11    ALTER COLUMN Count SET NOT NULL;
```

Листинг 3.23 – Создание ограничений целостности таблицы StudentBook

```
1 ALTER table diary.StudentBook
2     ADD CONSTRAINT fk_bookId foreign key(BookId) references
          diary.Library(BookId) ON DELETE CASCADE,
3     ADD CONSTRAINT fk_studentId foreign key(StudentId)
          references diary.Student(StudentId) ON DELETE CASCADE,
4     ALTER COLUMN StudentId SET NOT NULL,
5     ALTER COLUMN BookId SET NOT NULL;
```

Листинг 3.24 – Создание ограничений целостности таблицы TeacherSubject

```
1 ALTER table diary.TeacherSubject
2     ADD CONSTRAINT fk_TeacherId foreign key(TeacherId)
          references diary.Teacher(TeacherId) ON DELETE CASCADE,
3     ADD CONSTRAINT fk_SubjectId foreign key(SubjectId)
          references diary.Subject(SubjectId) ON DELETE CASCADE,
```

```
4     ALTER COLUMN TeacherId SET NOT NULL,
5     ALTER COLUMN SubjectId SET NOT NULL;
```

3.5 Создание триггера

В конструкторской части был разработан триггер для проверки валидности новой записи с помощью расширения PL/pgSQL. В листинге 3.25 представлено создание триггера и соответствующей функции.

Листинг 3.25 – Создание триггера

```
1  CREATE OR REPLACE FUNCTION check_book_constraints()
2  RETURNS TRIGGER AS $$
3  BEGIN
4      IF NEW.Count <= 0 THEN
5          RAISE EXCEPTION 'Quantity of books must be greater than
               0';
6      END IF;
7
8      IF NEW.DatePublication > CURRENT_DATE THEN
9          RAISE EXCEPTION 'Publication date must be before or
               equal to the current date';
10     END IF;
11
12     RETURN NEW;
13 END;
14 $$ LANGUAGE plpgsql;
15
16 CREATE TRIGGER book_constraints_trigger
17 BEFORE INSERT ON diary.Library
18 FOR EACH ROW
19 EXECUTE FUNCTION check_book_constraints();
```

3.6 Создание ролей базы данных

В конструкторской части были выделены 4 роли на уровне базы данных: гость, ученик, учитель и администратор. В листингах 3.26 — 3.29

представлено создание ролей базы данных в соответствии с ролевой моделью.

Листинг 3.26 – Создание роли гостя

```
CREATE ROLE guest WITH
    LOGIN
    NOSUPERUSER
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    PASSWORD 'guest'
    CONNECTION LIMIT -1;

GRANT USAGE ON SCHEMA diary TO guest;

GRANT SELECT ON diary.User,
                diary.UserRole,
                diary.Student,
                diary.Teacher
                TO guest;
```

Листинг 3.27 – Создание роли ученика

```
CREATE ROLE student WITH
    LOGIN
    NOSUPERUSER
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    PASSWORD 'student'
    CONNECTION LIMIT -1;

GRANT USAGE ON SCHEMA diary TO student;

GRANT SELECT ON diary.Class,
                diary.User,
                diary.UserRole,
                diary.Student,
                diary.Teacher,
                diary.Subject,
                diary.Schedule,
                diary.Assessment,
```

```
20                  diary.Library,
21                  diary.StudentBook
22                  TO student;
23
24 GRANT UPDATE ON diary.Library
25                  TO student;
26 GRANT INSERT, DELETE ON diary.StudentBook
27                          TO student;
```

Листинг 3.28 – Создание роли учителя

```
1  CREATE ROLE teacher WITH
2      LOGIN
3      NOSUPERUSER
4      NOCREATEDB
5      NOCREATEROLE
6      NOREPLICATION
7      PASSWORD 'teacher'
8      CONNECTION LIMIT -1;
9
10 GRANT USAGE ON SCHEMA diary TO teacher;
11
12 GRANT SELECT ON diary.Class,
13                  diary.User,
14                  diary.UserRole,
15                  diary.Teacher,
16                  diary.Student,
17                  diary.Subject,
18                  diary.Schedule,
19                  diary.Assessment,
20                  diary.TeacherSubject
21                  TO teacher;
22
23 GRANT INSERT, DELETE, UPDATE ON diary.Assessment
24                                  TO teacher;
```

Листинг 3.29 – Создание роли администратора

```
1  create role admin WITH
2      LOGIN
3      SUPERUSER
4      PASSWORD 'admin'
```

```
5     CONNECTION LIMIT -1;
```

3.7 Аутентификация и авторизация пользователя

В листинге 3.30 представлен пример аутентификации для ученика.

Листинг 3.30 – Аутентификация ученика

```
1  val query = "SELECT * FROM diary.user U " +
2          "JOIN diary.student T ON T.StudentId = U.UserId " +
3          "JOIN diary.userrole R ON U.userid = R.Userid " +
4          "WHERE Password = ? AND Identifier = ? AND R.roleid = 2"
5
6  var student: Student? = null
7
8  try {
9      dbConnector.getConnection()?.use { connection ->
10         val preparedStatement =
               connection.prepareStatement(query)
11         preparedStatement.setString(1, request.Password)
12         preparedStatement.setString(2, request.Identifier)
13
14         preparedStatement.executeQuery()?.use { resultSet ->
15             if (resultSet.next())
16                 student = StudentFormation(resultSet)
17         }
18     } ?: throw ConnectBDException()
19 } catch (e: Exception) {
20     println("Error creating user: ${e.message}")
21     throw SingInException()
22 }
23
24 if (student == null)
25     throw SingInException()
26
27 return get_student(student, 0)
```

В листинге 3.31 представлена авторизация пользователя.

Листинг 3.31 – Авторизация пользователя

```
fun changeUser(
        role: NameRole,
        user: String = null.toString(),
        password: String = null.toString(),
    ) {
        when (role) {
            NameRole.STUDENT -> {
                this.user = STUDENT_NAME
                this.password = STUDENT_PASSWORD
            }
            NameRole.TEACHER -> {
                this.user = TEACHER_NAME
                this.password = TEACHER_PASSWORD
            }
            NameRole.ADMIN -> {
                this.user = ADMIN_NAME
                this.password = ADMIN_PASSWORD
            }
            else -> {
                this.user = GUEST_NAME
                this.password = GUEST_PASSWORD
            }
        }
    }

private fun connect() {
        try {
            Class.forName("org.postgresql.Driver")
            connection = DriverManager.getConnection(url, user,
                password)
        } catch (e: Exception) {
            throw ConnectBDException()
        }
    }
```

3.8 Тестирование

Для тестирования проекта были реализованы модульные тесты для компонента доступа к данным и для компонента бизнес-логики. Для отдельных тестов запускается скрипт создания таблиц и в случае необходимости база данных заполняется тестовыми данными. После выполнения тестирования запускается скрипт удаления базы данных.

В листинге 3.32 представлены примеры тестирования компонента доступа к данным.

Листинг 3.32 – Модульные тесты для компонента доступа к данным

```
1  class ScheduleRepoTest {
2      @Test
3      fun createSchedule() {
4          var request_create_1 =
               CreateScheduleRequest(DayOfWeek.MONDAY, 2,
               GroupSchedule.JOINT, 1, 2, 10)
5          var sut = ScheduleRepo(Logger(), PostgresDBConnector())
6          var actual = sut.CreateSchedule(request_create_1)
7          assertEquals(0, actual)
8      }
9
10     @Test
11     fun getScheduleForTeacher() {
12         var request_get_1 = GetScheduleForTeacherRequest(1)
13         var sut = ScheduleRepo(Logger(), PostgresDBConnector())
14         var actual =
               sut.GetScheduleForTeacher(request_get_1).error
15         assertEquals(0, actual)
16     }
17
18     @Test
19     fun getScheduleForClass() {
20         var request_get_1 = GetScheduleForClassRequest(1)
21         var sut = ScheduleRepo(Logger(), PostgresDBConnector())
22         var actual = sut.GetScheduleForClass(request_get_1).error
23         assertEquals(0, actual)
24     }
```

Для тестирования компонента бизнес-логики также использовалась библеотека Mockito [21]. Она используется для имитации поведения компонента доступа к данным, что позволяет сосредоточиться на проверке бизнес-логики.

В листинге 3.33 представлены примеры тестирования компонента бизнес логики.

Листинг 3.33 – Модульные тесты для компонента бизнес-логики

```
1  class ClassServiceTest {
2      private  val mockClassRepository =
           Mockito.mock(ClassStorage::class.java)
3
4      @Test
5      fun getClass_OK() {
6          val request_1 = GetClassByIdRequest(1)
7
8          Mockito.`when`(mockClassRepository
9              .GetClassById(request_1))
10             .thenReturn(get_class(Class(1, "A",
                   ClassNumber.TWO), 0))
11         val sut = ClassService(Logger(), mockClassRepository)
12
13         val actual = sut.GetClassById(request_1).res_Class
14         val expected = Class(1, "A", ClassNumber.TWO)
15         assertEquals(expected, actual)
16     }
17
18     @Test
19     fun createClass_OK() {
20         val request_create_1 = CreateClassRequest(1, "E",
               ClassNumber.SEVEN)
21         val request_get_1 = GetClassByNameRequest("E",
               ClassNumber.SEVEN)
22
23         Mockito.`when`(mockClassRepository
24             .CreateClass(request_create_1)).thenReturn(0)
25         Mockito.`when`(mockClassRepository
26             .GetClassByName(request_get_1))
27             .thenReturn(get_class(null, 3))
28         val sut = ClassService(Logger(), mockClassRepository)
29
```

```
30          val actual = sut.CreateClass(request_create_1)
31          val expected = 0
32          assertEquals(expected, actual)
33      }
34      @Test
35      fun deleteClass_OK() {
36          val request_delete_1 = DeleteClassRequest(1)
37          val request_get_1 = GetClassByIdRequest(1)
38          Mockito.`when`(mockClassRepository
39              .GetClassById(request_get_1))
40              .thenReturn(get_class(Class(1, "E",
                    ClassNumber.NINE), 0)
41          )
42          Mockito.`when`(mockClassRepository
43              .DeleteClass(request_delete_1)).thenReturn(0)
44          val sut = ClassService(Logger(), mockClassRepository)
45
46          val actual = sut.DeleteClass(request_delete_1)
47          val expected = 0
48          assertEquals(expected, actual)
49      }
50  }
```

3.9 Пример пользовательского интерфейса

Примеры интерфейса для авторизации представлен на рисунке 3.1.

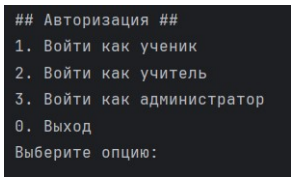

Рисунок 3.1 – Пример интерфейса для авторизации

Примеры интерфейса ученика представлен на рисунке 3.2.

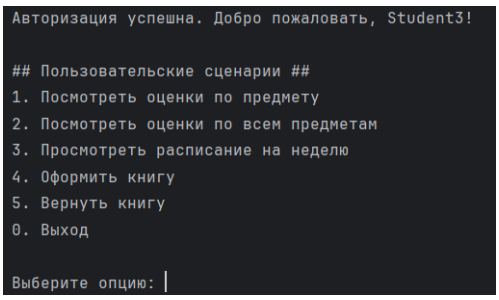

Рисунок 3.2 – Пример интерфейса для ученика

Примеры интерфейса учителя представлен на рисунке 3.3.

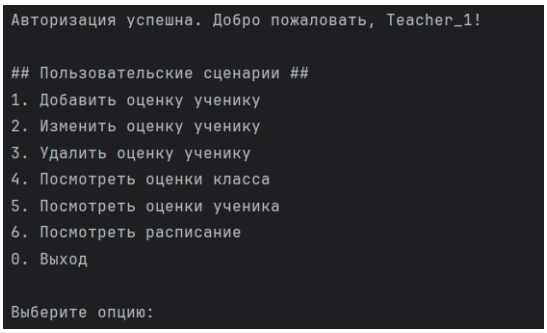

Рисунок 3.3 – Пример интерфейса для учителя

Примеры интерфейса администратора представлен на рисунке 3.4.

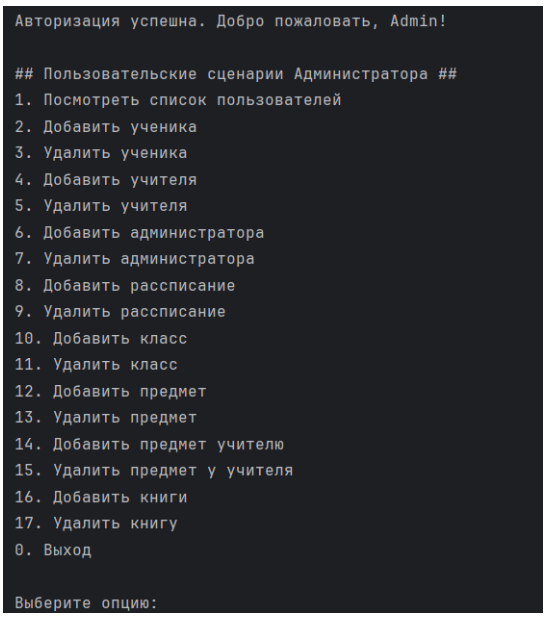

Рисунок 3.4 – Пример интерфейса для администратора

Вывод

В данном разделе выбраны и описаны средства реализации: в качестве системы управления базами данных выбрана PostgreSQL; язык программирования Kotlin. Приведены детали реализации ролей, триггера, создания таблиц базы данных и накладываемые на них ограничения целостностей, а также аунтификация и авторизация пользователя. Было описано тестирование разработанного функционала и предоставлен интерфейс для взаимодействия пользователя с базой данных.

4 Исследовательская часть

4.1 Технические характеристики

Исследование проводилось на устройстве со следующими техническими характеристиками:

- Операционная система: Windows 10 Pro 64-разрядная система версии 22H2 [22];
- Оперативная память: 8,00 ГБ;
- Процессор: Intel(R) Core(TM) i5-9300H CPU 2.40 ГГц [23].

4.2 Типы используемых индексов

В исследовании используются:

- кластеризованный индекс (КИ): определяет физический порядок хранения строк в таблице. Таблица может иметь только один кластеризованный индекс, так как строки могут быть упорядочены только одним способом. Строки таблицы физически сортируются в порядке кластеризованного индекса. Но при последующем обновлении таблицы изменения не кластеризуются, то есть не предпринимается попыток сохранить новые или обновленные строки в соответствии с порядком их индексации [24];
- некластеризованный индекс (НКИ): хранит указатели на строки данных в таблице. В таблице может быть множество не кластеризованных индексов. Данный тип индексов не влияет на физический порядок строк в таблице.

4.3 Описание исследования

Целью исследования является определение зависимости времени выполнения запроса от количества записай в таблицу и от запроса. Для проведения исследования выбрана таблица StudentBook, для индексации выбрано поле BookID, для первого запроса и таблица User, для индексации выбрано поле Birthday, для второго запроса.

Время выполнения запроса анализировалось с помощью встроенной в PostgreSQL команды EXPLAIN [25]. Для исследования было взято два запроса, показанные листинге 4.1 и 4.2.

Листинг 4.1 – Первый анализируемый запрос

```
1 SELECT * FROM diary.library as L1
2 JOIN diary.studentbook as S1 on L1.BookId = S1.BookId
3 WHERE S1.BookId = 5
```

Листинг 4.2 – Второй анализируемый запрос

```
1 SELECT * FROM diary.user
2 WHERE birthday < '2010-05-20';
```

Создание кластерного индекса для первого и второго запросов приведено в листингах 4.3 и 4.4 соответственно.

Листинг 4.3 – Кластеризация таблицы StudentBook по полю bookid

```
1 CREATE INDEX cluster_id ON diary.studentbook (bookid);
2 CLUSTER diary.studentbook USING cluster_index;
```

Листинг 4.4 – Кластеризация таблицы User по полю birthday

```
1 CREATE INDEX cluster_date ON diary.user (birthday);
2 CLUSTER diary.user USING cluster_date;
```

Так как кластеризация — это одноразовая операция, то перед тем, как замерять время выполнения запроса с кластеризованным индексом, необходимо повторно выполнить кластеризацию таблицы.

В листингах 4.5 и 4.6 приведена повторная кластеризация таблиц для первого и второго запроса соответственно.

Листинг 4.5 – Повторная кластеризация таблицы StudentBook

```
1 CLUSTER diary.studentbook;
```

Листинг 4.6 – Повторная кластеризация таблицы User

```
1 CLUSTER diary.user;
```

Создание некластерного индекса для первого и второго запросов приведено в листингах 4.7 — 4.8 соответственно.

Листинг 4.7 – Создание некластерного индекса по полю bookid

```
1 CREATE INDEX index_id ON diary.studentbook (bookid);
```

Листинг 4.8 – Создание некластерного индекса по полю birthday

```
1 CREATE INDEX index_date ON diary.user (birthday);
```

4.4 Результаты исследования

Результаты замеров времени (в миллисекундах) выполнения запроса в зависимости от количества записей и наличия индекса приведены в таблицах 4.1 и 4.2. При этом было проведено усреднение для получения наиболее точных результатов.

Таблица 4.1 – Сравнение времени выполнения первого запроса с различными индексами

Количество записей	Запрос без индексов, мс	КИ, мс	НКИ, мс
1	0.036	0.026	0.026
10	0.058	0.027	0.025
20	0.064	0.030	0.034
30	0.069	0.037	0.048
50	0.064	0.043	0.055
70	0.073	0.050	0.053
100	0.055	0.045	0.052
150	0.062	0.051	0.064
200	0.066	0.056	0.062
250	0.071	0.052	0.070
300	0.099	0.064	0.078
350	0.114	0.070	0.086
400	0.139	0.073	0.085
450	0.145	0.075	0.089
500	0.157	0.083	0.092
600	0.170	0.087	0.090
700	0.184	0.095	0.099
800	0.206	0.106	0.106
900	0.218	0.110	0.116
1000	0.233	0.123	0.120
10000	1.413	0.150	0.171
100000	6.066	0.582	0.611

На рисунке 4.1 приведены графические результаты сравнения времени выполнения первого запроса от количества записей (от 10 до 1000).

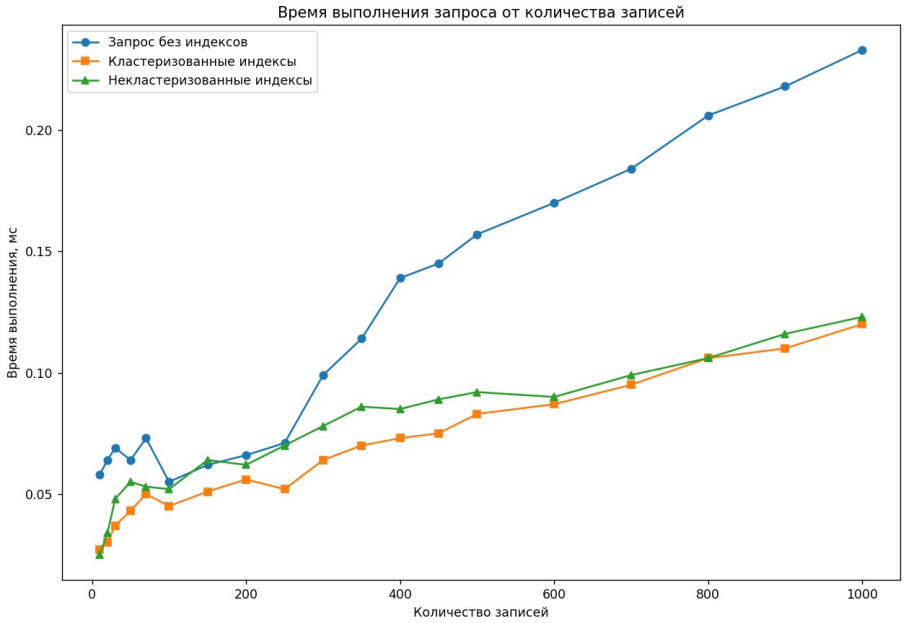

Рисунок 4.1 – Зависимость времени выполнения первого запроса от количества записей (от 10 до 1000) и наличия индексации

На рисунке 4.2 приведены графические результаты сравнения времени выполнения запроса от количества записей (от 1 до 100000).

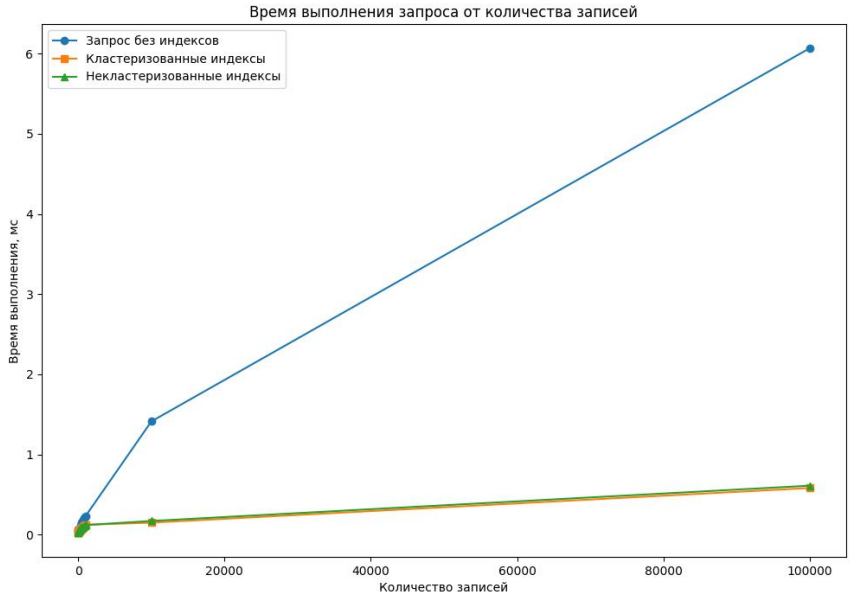

Рисунок 4.2 – Зависимость времени выполнения первого запроса от количества записей (от 1 до 100000) и наличия индексации

По результатам исследования видно, что время выполнения запроса

увеличивается с ростом количества записей. При 10000 записях в таблице запрос без индексов выполняет в 4.29 раз быстрее, чем такой же запрос при 100000 записей. Запрос с кластеризованным индексом при 10000 записях в таблице выполняется в 3.88 раз быстрее, чем такой же запрос при 100000 записей. А запрос с некластеризованным индексом при 10000 записях в таблице выполняет в 3.57 раз быстрее, чем такой же запрос при 100000 записей.

При 1000 записях в таблице запрос с кластеризованным индексом выпоняется быстрее запроса без индекса в 1.94 раз. А запрос с некластеризованным индексом выпоняется быстрее запроса без индекса в 1.89 раз. А при 100000 записях в таблице, запрос с кластеризованным индексом выпоняется быстрее запроса без индекса в 10.42 раз. А запрос с некластеризованным индексом выпоняется быстрее запроса без индекса в 9.93 раз.

Таблица 4.2 – Сравнение времени выполнения второго запроса с различными индексами

Количество записей	Запрос без индексов, мс	КИ, мс	НКИ, мс
1	0.027	0.020	0.022
10	0.035	0.029	0.031
100	0.055	0.039	0.040
1000	0.228	0.139	0.144
10000	1.142	0.569	0.625
100000	11.077	5.244	5.638

На рисунке 4.3 приведены графические результаты сравнения времени выполнения второго запроса от количества записей.

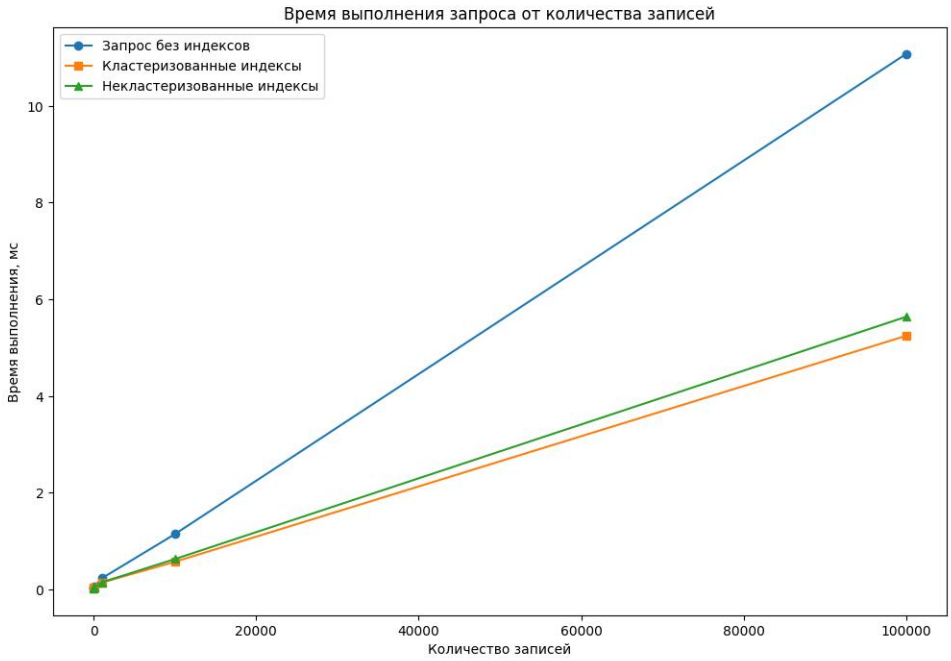

Рисунок 4.3 – Зависимость времени выполнения второго запроса от количества записей и наличия индексации

По результатам исследования второго запроса видно, что время выполнения запроса также увеличивается с ростом количества записей. При 10000 записях в таблице запрос без индексов выполняет в 9.69 раз быстрее, чем такой же запрос при 100000. Запрос с кластеризованным индексом при 10000 записях в таблице выполняется в 9.21 раз быстрее, чем такой же запрос при 100000 записей. А запрос с некластеризованным индексом при 10000 записях в таблице выполняет в 9.02 раз быстрее, чем такой же запрос при 100000 записей.

При 100000 записях в таблице, запрос с кластеризованным индексом выпоняется быстрее запроса без индекса в 2.11 раз. А запрос с некластеризованным индексом выпоняется быстрее запроса без индекса в 1.96 раз.

Первый запрос с кластеризованным индексом при 100000 записях в таблице выполняется быстрее второго запрос с кластеризованным индексом в 90.10 раз быстрее. Это связано с тем, то эффективность кластеризованных индексов зависит от структуры данных на которые они указывают.

Увеличение времени выполнения обычного запроса с увеличением количества записей в таблице связано с тем, что в случае отсутствия ин-

дексов, СУБД вынуждена выполнять полное сканирование таблицы для выполнения запроса. Это означает, что она последовательно проходит через каждую запись, что приводит к линейному росту времени выполнения запроса при увеличении количества записей.

Запрос с кластеризованным индексом в данном исследовании работает быстрее запроса с некластеризованным индексом, это связано с тем, что строки таблицы с кластеризованнм индексом физически хранятся на диске в том же порядке, что и индекс. А при некластеризованном индексе появляется второй список, содержащий указатели на физические строки. И при попытке получить все столбцы необходимо переходить сначала к индексу, а затем к таблице.

Вывод

В данном разделе было проведено исследование зависимости времени выполнения запросов при использования различных индексов от количества записей в таблице. В результате исследования было получено, что увеличение количества записей приводит к линейному росту времени выполнения запроса. Дополнительно было получено, что по времени эффективнее использовать кластерные индексы.

Так первый запрос при 10000 записях в таблице запрос без индексов выполняет в 4.29 раз быстрее, чем такой же запрос при 100000. А для второго запроса при 10000 записях в таблице запрос без индексов выполняет в 9.69 раз быстрее, чем такой же запрос при 100000.

Также при 100000 записях в таблице первый запрос с кластеризованным индексом выпоняется быстрее запроса без индекса в 10.42 раз, а тот же запрос с некластеризованным индексом выпоняется быстрее запроса без индекса в 9.93 раз. Запрос с кластеризованным индексом выпоняется быстрее запроса с некластеризованным индексом в 1.05 раз.

Для второго запроса при 100000 записях в таблице запрос с кластеризованным индексом выпоняется быстрее запроса без индекса в 2.11 раз. Тот же запрос с некластеризованным индексом выпоняется быстрее запроса без индекса в 1.96 раз и медленне запроса с кластеризованным индексом в 0.93 раз.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы была достигнута поставленная цель: разработана база данных для электронного дневника.

Для достижение цели были выполнены следующие задачи:

- проведен анализ предметной области электронного дневника;
- спроектирована базу данных, описаны сущности и проектируемая ролевая модель базы данных;
- разработана программная реализация приложения, описан интерфейс доступа к базе данных;
- проведено исследование зависимости времени выполнения запроса от количества записей в таблице.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Электронный дневник. Чем он лучше обычного? [Электронный ресурс]. — Режим доступа: https://nsportal.ru/shkola/inostrannye-yazyki/library/2017/03/26/elektronnyy-dnevnik-chem-on-luchshe-obychnogo (дата обращения: 02.04.2024).
2. Описание системы — NetSchool [Электронный ресурс]. — Режим доступа: http://netschool.school.ioffe.ru/prod_descr.htm (дата обращения: 02.04.2024).
3. SmileS.Школьная Карта — цифровая образовательная платформа, которая делает образование в России качественным и доступным! [Электронный ресурс]. — Режим доступа: https://shkolnaya-karta.ru/uchenikam.html (дата обращения: 02.04.2024).
4. АВЕРС: Электронный Классный Журнал [Электронный ресурс]. — Режим доступа: https://cit-avers.ru/produktsiya/shkola/ias-avers-elektronnyj-klassnyj-zhurnal/ (дата обращения: 02.04.2024).
5. Сравнение различных электронных дневников для школ [Электронный ресурс]. — Режим доступа: https://nsportal.ru/shkola/administrirovanie-shkoly/library/2012/11/19/sravnenie-razlichnykh-elektronnykh-dnevnikov-dlya (дата обращения: 02.04.2024).
6. К. Дж. Дейт. «Введение в системы баз данных», 8-е издание, издательский дом «Вильяме», 2005. — 1238 с.
7. Томас Коннолли, Каролин Бегг. «Базы данных: Проектирование, реализация и сопровождение. Теория и практика», 3-е издание, издательский дом «Вильяме», 2017. — 1440 с.
8. What is a Relational Database (RDBMS)? [Электронный ресурс]. — Режим доступа: https://www.oracle.com/database/what-is-a-relational-database/ (дата обращения: 05.04.2024).
9. TECHNICAL SCIENCE / «Colloquium-journal» 2(54),2020, Васильева К.Н., Хусаинова Г.Я, Реляционные базы данных, 2020.

10. What is ACID Compliance? [Электронный ресурс]. — Режим доступа: https://www.mongodb.com/databases/acid-compliance (дата обращения: 05.04.2024).

11. What Is an In-Memory Database? | AWS Amazon [Электронный ресурс]. — Режим доступа: https://aws.amazon.com/ru/nosql/in-memory/ (дата обращения: 10.04.2024).

12. ТОП-10 САМЫХ ПОПУЛЯРНЫХ СИСТЕМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ [Электронный ресурс]. — Режим доступа: https://mixtelecom.ru/blog/top-10-samyh-populyarnyh-sistem-upravleniya-bazami-dannyh (дата обращения: 10.04.2024).

13. СУБД Oracle [Электронный ресурс]. — Режим доступа: https://www.oracle.com/cis/database/ (дата обращения: 10.04.2024).

14. MySQL [Электронный ресурс]. — Режим доступа: https://www.mysql.com/ (дата обращения: 10.04.2024).

15. PostgreSQL: The World's Most Advanced Open Source Relational Database [Электронный ресурс]. — Режим доступа: https://www.postgresql.org/ (дата обращения: 10.04.2024).

16. Oracle, MySQL, PostgreSQL, SQLite, SQL Server: Performance based competitive analysis [Электронный ресурс]. — Режим доступа: http://dspace.daffodilvarsity.edu.bd:8080/handle/123456789/4437 (дата обращения: 10.04.2024).

17. Kotlin Programming Language [Электронный ресурс]. — Режим доступа: https://kotlinlang.org/ (дата обращения: 16.04.2024).

18. Ведущая IDE для разработки на Java и Kotlin [Электронный ресурс]. — Режим доступа: https://www.jetbrains.com/ru-ru/idea/ (дата обращения: 16.04.2024).

19. Kotlin SHA-256: Secure Hashing Methods Explained with Code Examples [Электронный ресурс]. — Режим доступа: https://markscodenotes.com/kotlin-sha-256-secure-hashing-methods-explained-with-code-examples/ (дата обращения: 20.04.2024).

20. PreparedStatement [Электронный ресурс]. — Режим доступа: https://developer.android.com/reference/kotlin/java/sql/PreparedStatement (дата обращения: 16.04.2024).

21. JUnit и фреймворк Mockito [Электронный ресурс]. — Режим доступа: https://java-online.ru/junit-mockito.xhtml (дата обращения: 20.04.2024).

22. Windows [Электронный ресурс]. — Режим доступа: https://www.microsoft.com/ru-ru (дата обращения: 20.04.2024).

23. Intel [Электронный ресурс]. — Режим доступа: https://www.intel.com/content/www/us/en/support/ru-banner-inside.html (дата обращения: 20.04.2024).

24. PostgreSQL: CLUSTER – Improve Index Performance (No default cluster index) [Электронный ресурс]. — Режим доступа: https://www.dbrnd.com/2016/12/postgresql-cluster-improve-index-performance-no-default-cluster-index-explicit-lock-physical-order-data/ (дата обращения: 23.04.2024).

25. Использование EXPLAIN [Электронный ресурс]. — Режим доступа: https://postgrespro.ru/docs/postgrespro/9.5/using-explain (дата обращения: 20.04.2024).

ПРИЛОЖЕНИЕ А

Презентация к курсовой работе

Презентация содержит 13 слайдов.