СОДЕРЖАНИЕ

ВВЕДЕНИЕ **5**

1 Аналитическая часть **6**

1.1 Анализ существующих решений 6

1.2 Формализация данных 6

1.3 Формализация ролей 9

1.4 Формализация задачи 9

1.5 Анализ баз данных 10

1.6 Вывод из аналитической части 12

2 Конструкторская часть **13**

2.1 Диаграмма проектируемой базы данных 13

2.2 Описание сущностей базы данных 15

2.3 Ролевая модель 18

2.4 Триггер базы данных 18

2.5 Вывод из конструкторской части 19

3 Технологическая часть **20**

3.1 Выбор системы управления базами данных 20

3.2 Выбор объектного хранилища 20

3.3 Средства реализации 21

3.4 Создание таблиц базы данных 22

3.5 Создание ролей базы данных 22

3.6 Реализация триггера 23

3.7 Интерфейс доступа к базе данных 24

3.8 Тестирование разработанного функционала 25

3.9 Вывод из технолгической части 25

4 Исследовательская часть **26**

4.1 Технические характеристики 26

4.2 Постановка исследования 26

4.3 Результаты исследования 27

4.4 Вывод из исследовательской части 30

ЗАКЛЮЧЕНИЕ 31

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 34

ПРИЛОЖЕНИЕ А 35

Введение

В современном мире электронные образовательные платформы становятся все более популярными и широко используются для обучения и самосовершенствования. Автоматизация анализа информации об обучающихся, курсах, материалах и результатов обучения на таких платформах играет ключевую роль в обеспечении эффективного функционирования системы. В связи с увеличением количества создаваемых обучающих материалов, актуальной задачей становится разработка системы управления образовательной деятельностью, а также обеспечение безопасности и целостности данных.

Цель данной работы – разработка базы данных электронной образовательной платформы.

Чтобы достичь поставленной цели, требуется решить следующие задачи:

— проанализировать существующие решения;

— спроектировать базу данных;

— спроектировать приложение доступа к базе данных;

— реализовать приложение доступа к базе данных;

— исследовать зависимость времени проверки целостности данных на уровне приложения и базы данных от количества записей в таблице.

1 Аналитическая часть

В данной части проводится анализ существующих решений, формализация данных, ролей и задачи, анализ баз данных.

1.1 Анализ существующих решений

На рынке онлайн образования представлено множество платформ, предоставляющих образовательные материалы разной тематики и формы. Ключевым критерием, по которому были отобраны существующие решения, является наличие на платформе как теории, так и практики. При анализе отобранных решений были выделены следующие критерии:

— наличие подробного отчета о пройденном обучении;

— возможность создать и опубликовать на платформе собственный курс;

— понятный и простой пользовательский интерфейс;

— возможность основать собственную школу.

Таблица 1 – Сравнение существующих решений

Решение	Подробный отчет	Создание курса	Собственная школа	Простой польз. интерфейс
Stepik	+	+	-	+
Coursera	-	-	-	-
Яндекс Практикум	+	-	-	+

Из таблицы 1 видно, что существующие решения не удовлетворяют описанным требованиям в полной мере.

1.2 Формализация данных

В предметной области онлайн образования ключевыми сущностями являются школа и курс. Школа в данной работе включает в себя множество преподавателей, студентов и курсов, которые могут быть реализованы преподавателями данной школы. Также школа содержит дополнительную информацию о создателе, платежные данные для оплаты того или иного курса и прочую дополнительную информацию.

Каждый курс разбит на уроки, которые могут быть нескольких типов: текстовые, видео-уроки и практические. Текстовые и видео уроки предоставляют теоретические знания, в то время как практические задания реализуются в виде тестов с множественным выбором.

База данных должна хранить информацию о следующих сущностях:

— пользователи;

— школы;

— курсы;

— уроки, принадлежащие некоторому курсу;

— практические тесты;

— отзывы;

— сертификаты.

На рисунке 1 приведена ER-диаграмма сущностей в нотации Чена.

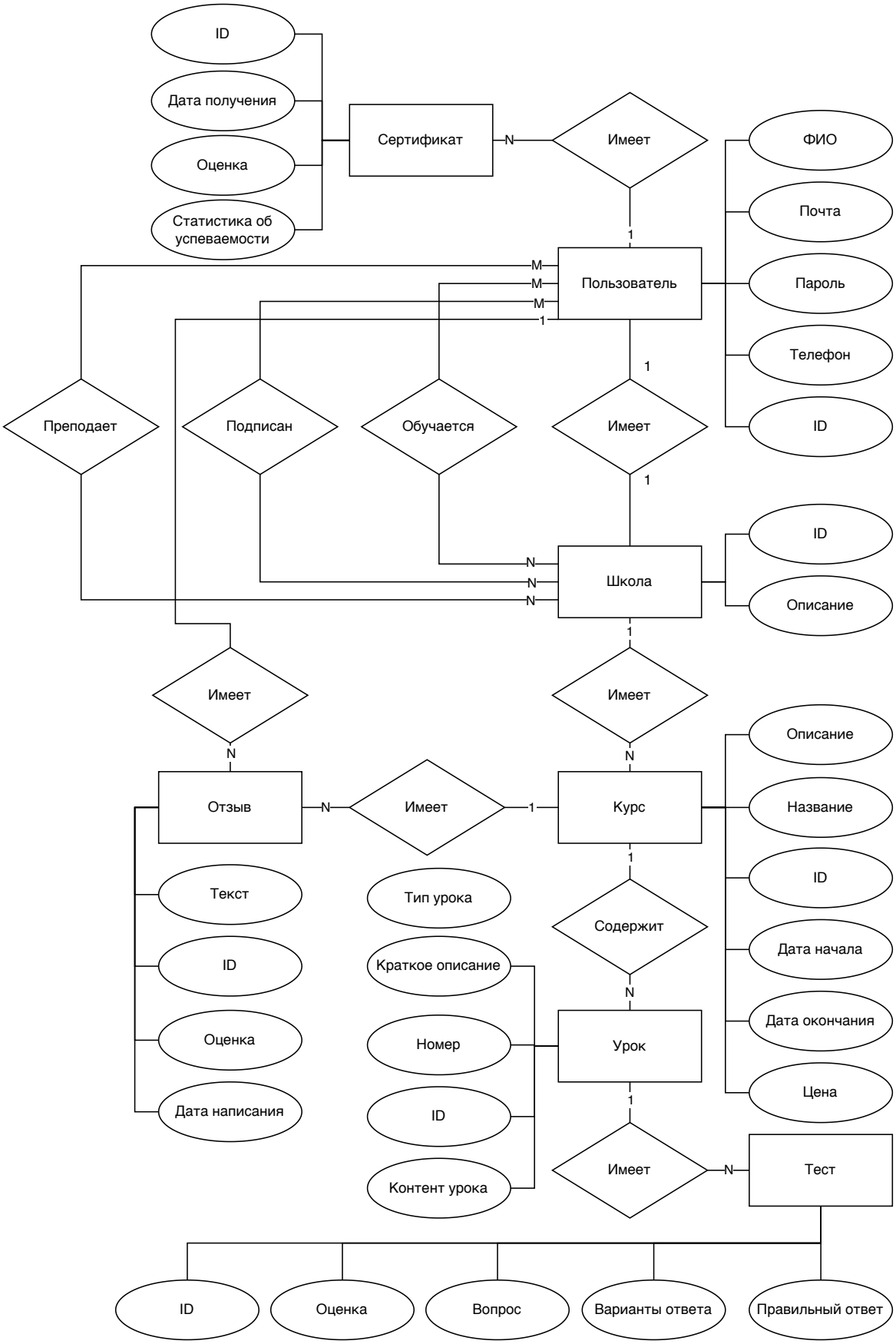

Рисунок 1 – ER-диаграмма сущностей в нотации Чена

1.3 Формализация ролей

В данной работе выделяются следующие роли пользователей:

— гость – неавторизованный пользователь, который может посматривать информацию о курсе, авторизоваться, зарегистрироваться;

— пользователь – авторизованный пользователь, может просматривать курсы, проходить, оплачивать их, по завершении курса получать сертификат и отчет об успеваемости и оставлять отзывы, стать преподавателем, создать собственную школу и курсы в ней, а также присоединиться к существующей школе в качестве преподавателя;

— администратор – авторизованный пользователь, может создавать, изменять и удалять пользователей, курсы, школы, уроки, отзывы.

1.4 Формализация задачи

Необходимо спроектировать базу данных для хранения информации о студентах и преподавателях, школах, курсах и уроках, входящих в состав курса, а также об отзывах пользователей. Разработанное приложение для доступа к базе данных должно предоставлять каждому пользователю возможность покупать и проходить курсы различной тематики. Также у каждого пользователя должна быть возможность создать собственную школу и образовательную программу или присоединиться к уже существующей школе в качестве преподавателя или составителя курса.

По прохождении курса пользователь должен иметь возможность получить сертификат о завершении обучения, включающий подробное описание его успеваемости и итоговую оценку. Диаграмма вариантов использования приведена на рисунке 2.

Рисунок 2 – Диаграмма вариантов использования

1.5 Анализ баз данных

База данных – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

СУБД – совокупность языковых и программных средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных [**williams**].

Модель данных представляет собой абстрактное, логическое описание элементов, таких как объекты и операции, которые в совокупности образуют виртуальную систему управления данными, с которой взаимодействует пользователь. Объекты модели описывают структуру данных, в то время как операции отражают их поведение.

Базы данных можно классифицировать по модели хранения на несколько типов:

— дореляционные модели данных — включают иерархические, сетевые модели и системы, использующие инвертированные списки.

— реляционные модели данных — основываются на таблицах, что делает их более гибкими и простыми в использовании.

— нереляционные модели данных — ориентированы на расширенные возможности работы с данными, выходящие за рамки реляционной парадигмы.

Дореляционные модели

Дореляционные базы данных представляют собой тип баз данных, который не использует табличную модель данных. Вместо этого, они хранят данные в структурах, таких как деревья, графы или объекты.

Реляционная модель данных

Реляционные базы данных основаны на реляционной модели данных, где данные организованы в виде таблиц, которые имеют связи между собой. Каждая таблица представляет отдельную сущность, а связи между таблицами устанавливаются с помощью ключевых полей. Важным свойством реляционных баз данных является их способность удовлетворять требованиям ACID [1]: атомарность, согласованность, изоляция, устойчивость. Примерами реляционных баз данных являются MySQL [2], PostgreSQL [3], Oracle [4] и SQL Server [5].

Нереляционная модель данных

Постреляционные базы данных – это базы данных, которые разработаны для обработки и анализа больших объемов данных. В отличие от реляционных

баз данных, основанных на таблицах, нереляционные базы данных предлагают более гибкие способы хранения данных, что делает их подходящими для определенных типов задач, таких как обработка больших объемов данных, работа с распределенными системами и требование высокой производительности. Примерами постреляционных баз данных являются Apache Hadoop [6], Apache Cassandra [7] и Apache Spark [8].

1.6 Вывод из аналитической части

С учетом 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 данной части будет представлена схема проектируемой базы данных, будут описаны сущности базы данных, ограничения целостности, ролевая модель и используемые функции и триггеры.

2.1 Диаграмма проектируемой базы данных

На рисунке 3 представлена диаграмма проектируемой базы данных.

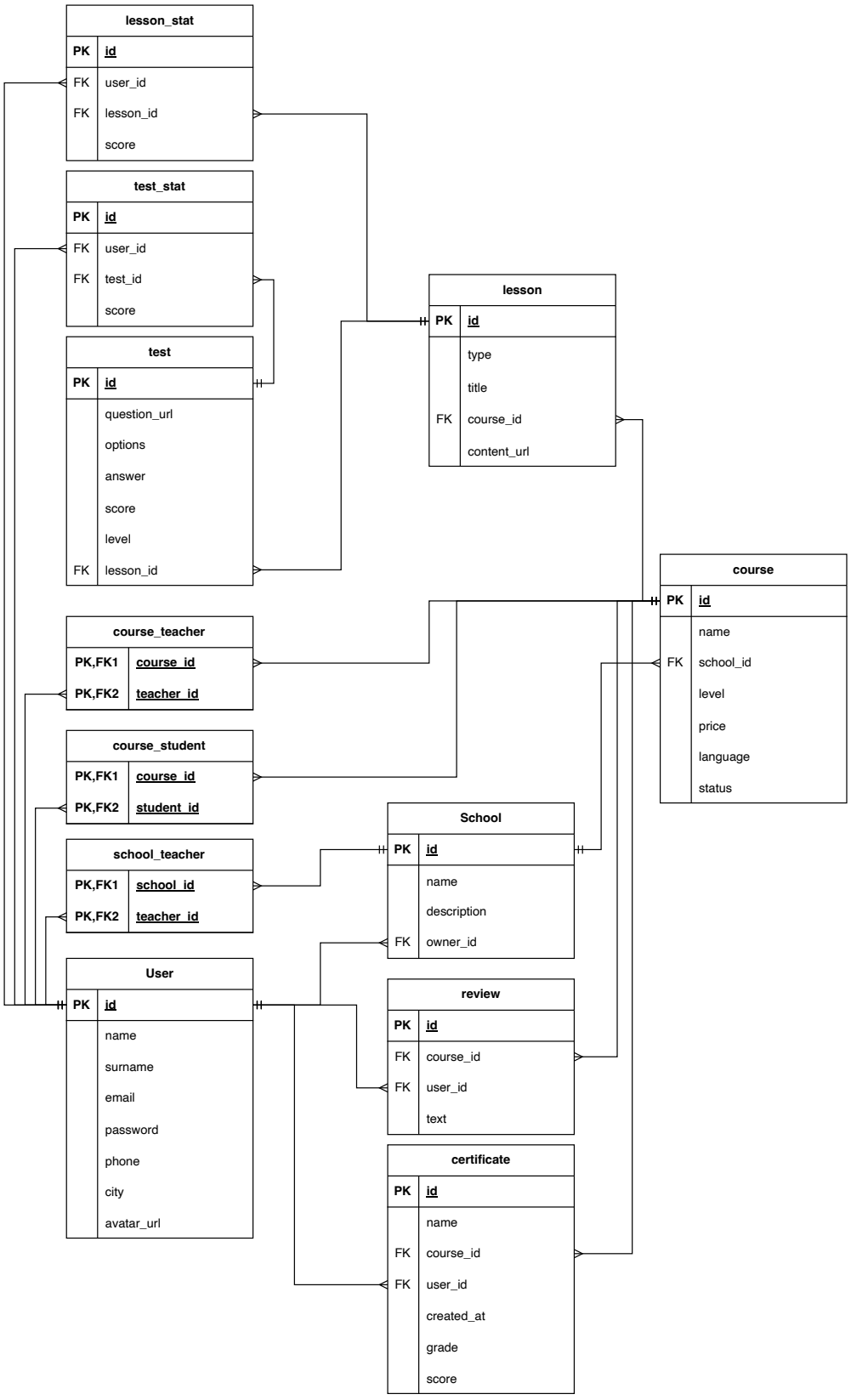

Рисунок 3 – Диаграмма проектируемой базы данных

2.2 Описание сущностей базы данных

Таблица 2 – Поля таблицы user

Поле	Тип данных	Ограничения целостности	Описание
id	uuid	primary key	Идентификатор пользователя
email	varchar(255)	unique not null	Почта пользователя
password	varchar(255)	not null	Пароль пользователя
name	varchar(255)	not null	Имя пользователя
surname	varchar(255)	not null	Фамилия пользователя
phone	varchar(255)	нет	Номер телефона пользователя
city	varchar(255)	нет	Город пользователя
avatar_url	text	нет	Ссылка на аватар пользователя

Таблица 3 – Поля таблицы school

Поле	Тип данных	Ограничения целостности	Описание
id	uuid	primary key	Идентификатор школы
name	varchar(255)	not null	Название школы
description	text	not null	Описание школы
owner_id	uuid	not null	Идентификатор владельца школы

Таблица 4 – Поля таблицы course

Поле	Тип данных	Ограничения целостности	Описание
id	uuid	primary key	Идентификатор курса
name	varchar(255)	not null	Название курса
school_id	uuid	not null	Идентификатор школы, владеющей курсом
level	int	not null	Уровень сложности курса
price	bigint	not null	Цена курса
language	varchar(255)	not null	Язык материала курса
status	course_status	not null	Текущий статус курса
rating	float	not null	Пользовательская средняя оценка курса

Поле status таблицы school имеет перечисляемый тип данных и может принимать три варианта: «draft», «ready», «published». Статус курса – текущее состояние готовности и наполнения материалом курса. Черновик курса может редактироваться автором, после чего может быть переведен в состояние готовности к публикации. После успешной публикации курса, состояние меняется на «опубликован».

Таблица 5 – Поля таблицы lesson

Поле	Тип данных	Ограничения целостности	Описание
id	uuid	primary key	Идентификатор урока
title	varchar(255)	not null	Название урока
type	lesson_type	not null	Тип урока
score	int	not null	Максимальная оценка за урок
theory_url	text	нет	Ссылка на теоретический материал урока
video_url	text	нет	Ссылка на видео материал урока
course_id	uuid	not null	Идентификатор курса, который содержит урок

Урок может быть трех видов: теоретический, видео-урок и практический урок. Тип урока описывается полем type таблицы lesson и может принимать значения: «theory», «video», «practice».

Таблица 6 – Поля таблицы test

Поле	Тип данных	Ограничения целостности	Описание
id	uuid	primary key	Идентификатор теста
task_url	text	not null	Ссылка на описание задачи теста
options	text	not null	Возможные варианты ответа
answer	text	not null	Ответ на задание
score	int	not null	Максимальная оценка за прохождение теста
level	int	not null	Уровень сложности теста
lesson_id	uuid	not null	Идентификатор урока, содержащего тест

Таблица 7 – Поля таблицы review

Поле	Тип данных	Ограничения целостности	Описание
id	uuid	primary key	Идентификатор отзыва об уроке
text	text	not null	Текст отзыва об уроке
course_id	uuid	not null	Идентификатор курса
user_id	uuid	нет	Идентификатор пользователя
rating	int	нет	Оценка курса

Таблица 8 – Поля таблицы certificate

Поле	Тип данных	Ограничения целостности	Описание
id	uuid	primary key	Идентификатор отзыва об уроке
name	varchar(1024)	not null	Название сертификата
score	int	not null	Итоговая оценка прохождения курса
grade	certificate_grade	not null	Степень сертификата
created_at	timestamp	not null	Дата выдачи сертификата
user_id	uuid	not null	Идентификатор пользователя
course_id	uuid	нет	Идентификатор курса

Сертификаты могут быть трех видов: бронзовый, серебряный и золотой. Тип сертификата описывается полем grade таблицы certificate и может принимать значения: «bronze», «silver», «gold».

Таблица 9 – Поля таблицы lesson_stat

Поле	Тип данных	Ограничения целостности	Описание
id	uuid	primary key	Идентификатор записи о результате прохождения урока
score	int	not null	Оценка прохождения урока
user_id	uuid	not null	Идентификатор пользователя
lesson_id	uuid	not null	Идентификатор урока

Таблица 10 – Поля таблицы test_stat

Поле	Тип данных	Ограничения целостности	Описание
id	uuid	primary key	Идентификатор записи о результате прохождения урока
score	int	not null	Оценка прохождения урока
user_id	uuid	not null	Идентификатор пользователя
test_id	uuid	not null	Идентификатор тест

Таблицы lesson_stat и test_stat хранят данные о прохождении пользователем уроков и тестов соответственно. Сертификат о прохождении курса формируется на основании информации из этих таблиц.

Таблица 11 – Поля таблицы course_student

Поле	Тип данных	Ограничения целостности	Описание
student_id	uuid	not null	Идентификатор пользователя
course_id	uuid	not null	Идентификатор курса

Таблица 12 – Поля таблицы course_teacher

Поле	Тип данных	Ограничения целостности	Описание
teacher_id	uuid	not null	Идентификатор пользователя
course_id	uuid	not null	Идентификатор курса

Таблица 13 – Поля таблицы school_teacher

Поле	Тип данных	Ограничения целостности	Описание
teacher_id	uuid	not null	Идентификатор пользователя
school_id	uuid	not null	Идентификатор школы

Таблица course_student является развязочной для курсов и обучающихся на них студентах. Развязочная таблица course_teacher связывает курсы и их преподавателей, а таблица school_teacher является развязочной для школ и их преподавателей.

2.3 Ролевая модель

В данной работе выделяются следующие роли:

— гость – имеет возможность просматривать курсы, школы и их преподавателей, отзывы о курсах;

— пользователь – имеет права на просмотр, редактирование и удаление собственных курсов, школ, уроков, тестов, а также может проходить приобретенные курсы;

— администратор – имеет все права для действий со всеми таблицами.

2.4 Триггер базы данных

К каждому курсу пользователи, прошедшие данный курс, могут оставить отзыв и выставить оценку качества. При этом средняя оценка курса должна меняться при операциях над таблицей отзывов. Для этого используется триггер базы данных, который сработает при добавлении, удалении или обновлении отзыва пользователя. Таким образов, средняя оценка пересчитывается и при получении информации о курсе исчезает необходимость пересчитывать ее каждый раз.

На рисунке 4 представлена схема алгоритма триггера для пересчета средней оценки курса.

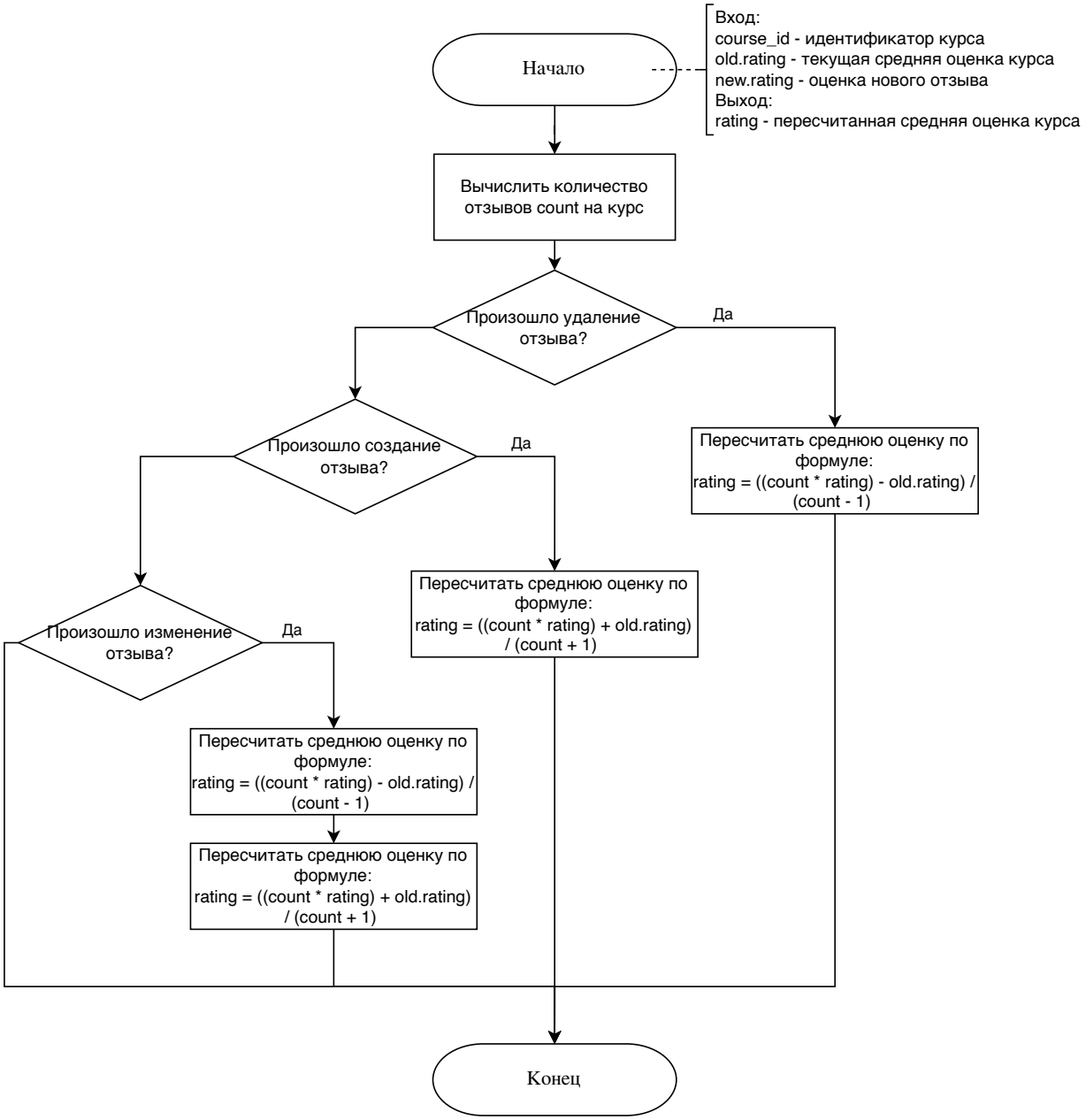

Рисунок 4 – Схема алгоритма триггера

2.5 Вывод из конструкторской части

В данной части были спроектирована схема базы данных, описаны сущности базы данных, ограничения целостности, ролевая модель и используемые функции и триггеры.

3 Технологическая часть

В данной части приводится обоснование выбора СУБД и средств реализации приложения для взаимодействия с базой данных. Также приводится реализация создания таблиц базы данных, реализация ролевой модели на уровне базы данных и реализация триггера. Кроме того описаны методы тестирования разработанного функционала и приведена реализация интерфейса доступа к базе данных.

3.1 Выбор системы управления базами данных

Чтобы выбрать СУБД необходимо рассмотреть следующие критерии:

— бесплатное и открытое программное обеспечение;

— поддержка процедурных расширений языка SQL;

— наличие личного опыта работы.

Таблица 14

Система управления базами данных	Бесплатное и открытое программное обеспечение	Поддержка процедурных расширений языка SQL	Наличие личного опыта работы
PostgreSQL [3]	+	+	+
Oracle [4]	-	+	-
SQLite [9]	+	+	-

По таблице 14 можно сделать вывод, что наиболее подходящей СУБД для данной работы является PostgreSQL.

3.2 Выбор объектного хранилища

Объектное хранилище — это тип системы хранения данных, предназначенный для управления большими объемами неструктурированной информации, такой как мультимедийные файлы, бэкапы, лог-файлы, документы и другие типы данных. Вместо традиционной файловой системы или реляционной базы данных, объектное хранилище организует данные в виде «объектов».

Для хранения файлов с материалами создаваемых преподавателями курсов необходимо использовать объектное хранилище. При этом в базе дан-

ных должны храниться ссылки на получение данных файлов пользователем курса.

Amazon S3 [10] — одно из самых популярных облачных хранилищ с поддержкой API, множеством классов хранения и глобальной инфраструктурой для высокой доступности. Данное хранилище не доступно на территории РФ, поэтому использование его в данной работе невозможно.

Google Cloud Storage [11] — это облачная служба хранения данных от Google, предназначенная для хранения и управления большими объёмами неструктурированных данных. Это объектное хранилище, которое позволяет загружать, сохранять и извлекать данные любого типа, например, мультимедийные файлы, резервные копии, журналы, архивы, документы и многое другое. Данное хранилище доступно в РФ, однако услуги, предоставляемые компанией платные.

MinIO [12] — это программное обеспечение с открытым исходным кодом, которое позволяет создавать объектные хранилища, совместимые с API Amazon S3. MinIO часто используется для развертывания облачных хранилищ на локальных серверах, в частных или гибридных облаках.

Учитывая доступность объектных хранилищ на территории РФ, их стоимость и наличие личного опыта работы, было выбрано хранилище MinIO для взаимодействия с файлами материалов курсов преподавателей.

3.3 Средства реализации

В данной работе использованы следующие средства реализации:

— язык программирования Golang [13];

— СУБД PostgreSQL [3];

— расширение языка SQL PL/pgSQL [14];

— автоматизация развертывания реализована с помощью Docker [15].

Доступ к базе данных реализован в виде консольного интерфейса. Тестирование функционала проводилось с помощью встроенной программы go test [16].

3.4 Создание таблиц базы данных

В листинге А.1 приведена реализация создания таблиц для сущностей предметной области.

3.5 Создание ролей базы данных

В листинге А.2 приведена реализация создания ролей базы данных.

3.6 Реализация триггера

В листинге 3.1 приведена реализация триггера для пересчета средней оценки курса.

Листинг 3.1 – Реализация триггера

```
create or replace function add_course_review()
    returns trigger as
$$
declare
    course_reviews_count numeric;
begin
    select count(*) from review where course_id=new.course_id
       into course_reviews_count;
    if (tg_op = 'DELETE') then
        update course set rating=((course_reviews_count *
           rating) - old.rating) /
             (course_reviews_count - 1) where id=old.course_id;
    elsif (tg_op = 'INSERT') then
        update course set rating=((course_reviews_count *
           rating) + new.rating) /
             (course_reviews_count + 1) where id=new.course_id;
    elsif (tg_op = 'UPDATE') then
        update course set rating=((course_reviews_count *
           rating) - old.rating) /
             (course_reviews_count - 1) where id=new.course_id;
        update course set rating=((course_reviews_count *
           rating) + new.rating) /
             (course_reviews_count + 1) where id=new.course_id;
    end if;
    return null;
end
$$ language plpgsql;

create trigger update_course
    after insert or update or delete
    on review for each row execute function add_course_review();
```

3.7 Интерфейс доступа к базе данных

В листинге 3.2 приведен пример пользовательского интерфейса.

Листинг 3.2 – Консольный интерфейс приложения

```
--------------------------------
Menu
0  Exit
1  Print menu
2  Sign In
3  Sign Up
4  Logout
5  Get all users
6  Get user account
7  Update user account
8  Get user purchased courses
9  Add free course to user library
10 Get all courses
11 Create course
12 Update course
13 Delete course
14 Get course lessons
15 Create course lesson
16 Get course teachers
17 Add course teacher
18 Publish course
19 Get all schools
20 Create school
21 Update school
22 Get school courses
23 Get school teachers
24 Add school teacher
25 Pass course lesson
26 Get lesson progress
27 Find course reviews
28 Add course review
29 Get course certificate
30 Generate course certificate
--------------------------------
Choose menu option:
```

3.8 Тестирование разработанного функционала

В процессе тестирования были использованы два подхода: модульное тестирование и интеграционное. Модульное тестирование проводилось с помощью стандартного пакета testing [16] языка golang. Для изоляции сервисного слоя приложения использовался подход mock тестирования [17] и библиотека Mockery [18].

Интеграционное тестирование уровня доступа к базе данных проводилось с использованием библиотеки Testcontainers [19].

На рисунке 5 изображены результаты пройденных тестов и их количество.

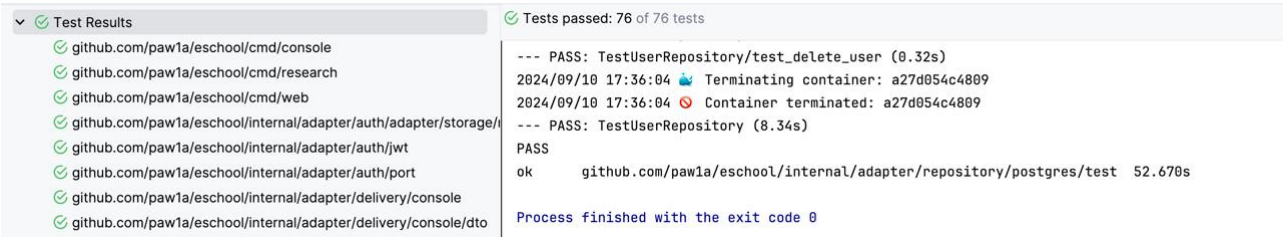

Рисунок 5 – Результаты пройденных тестов разработанного функционала

3.9 Вывод из технолгической части

В данной части была выбрана используемая СУБД и средства реализации приложения для взаимодействия с базой данных, приведена реализация создания таблиц базы данных, реализация ролевой модели на уровне базы данных и триггера. Были представлены методы тестирования разработанного функционала и пример реализации интерфейса доступа к базе данных.

4 Исследовательская часть

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

— операционная система MacOS Monterey 12.3;

— 16 ГБ оперативной памяти;

— процессор Apple M1 [20].

Тестирование проводилось на ноутбуке, включенном в сеть электропитания. Во время тестирования ноутбук был нагружен только встроенными приложениями окружения, окружением, а также непосредственно системой тестирования.

4.2 Постановка исследования

Целью исследования является нахождение зависимости времени проверки целостности данных на уровне приложения и базы данных от количества записей в таблице.

Курс может иметь несколько состояний в процессе подготовки материала преподавателем. Черновик курса может редактироваться преподавателем, после чего он должен быть переведен в состояние готовности. Для этого необходимо проверить, что курс удовлетворяет следующим требованиям:

— в курсе присутствует один или более теоретический или видео урок;

— в курсе присутствует один или более практический урок;

— все практический уроки курса должны иметь хотя бы один тест;

— максимальная оценка, выставляемая за прохождение урока, всех уроков должна быть больше нуля.

Данные проверки целостности курса могут быть проведены как на уровне базы данных, так и на уровне приложения. Необходимо сравнить временные характеристики выполнения для обоих способов.

Исследование проводится на случайных данных, при этом количество курсов в таблице равно 10, для каждого из которых необходимо провести проверку целостности. Замеры времени проводятся для двух диапазонов данных. Замеряется время проверки целостности курса для количества уроков от 100 до 2000 и от 1000 до 10000. Количество тестов в практических уроках фиксировано и равно 10. Тестовые данные генерируются с помощью библиотеки Faker [21].

4.3 Результаты исследования

В таблице 15 приведены результаты замеров времени выполнения реализации проверки целостности курса для количества уроков в курсе от 1000 до 10000 штук.

Таблица 15 – Результат замеров времени проверки целостности для количества записей от 1000 до 10000 с шагом 1000.

Количество уроков в курсе, шт	Время выполнения проверок целостности на уровне приложения, мс	Время выполнения проверок целостности на уровне базы данных, мс
1000	60	15
2000	219	85
3000	616	229
4000	899	339
5000	1102	456
6000	713	265
7000	2457	1080
8000	1603	684
9000	3467	1431
10000	1040	437

По таблице 15 был построен график 6.

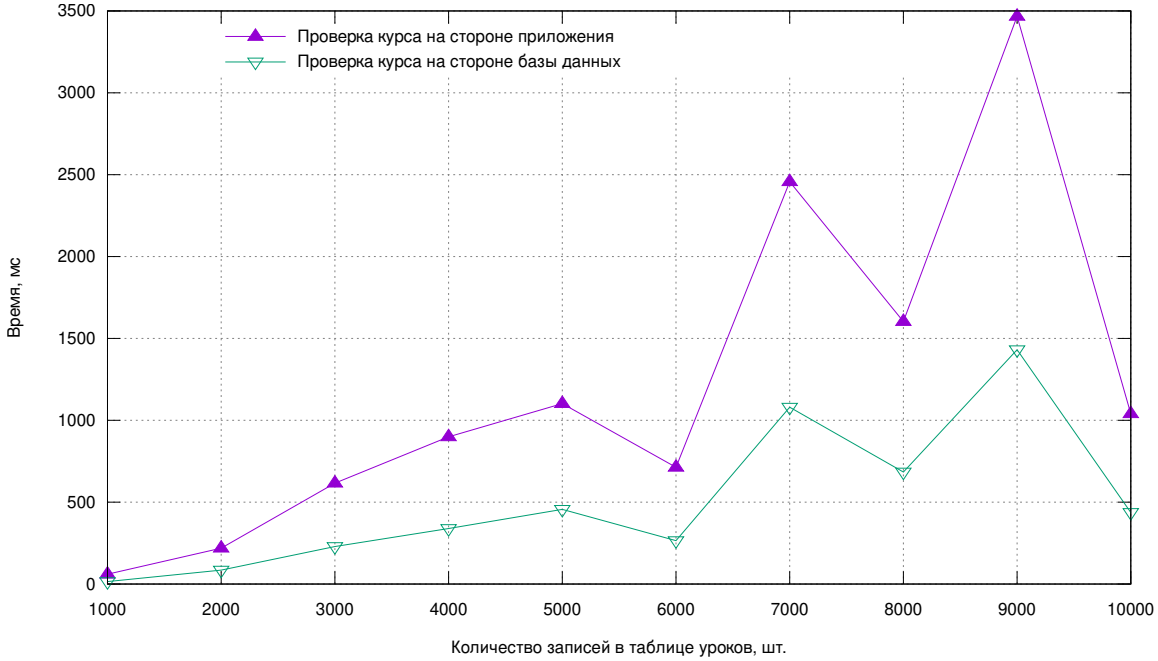

Рисунок 6 – График зависимости времени проверок целостности от количества уроков в курсе при диапазоне значений от 1000 до 10000

В таблице 16 приведены результаты замеров времени выполнения реализации проверки целостности курса для количества уроков в курсе от 1000 до 10000 штук.

Таблица 16 – Результат замеров времени проверки целостности для количества записей от 100 до 2000 с шагом 100.

Количество уроков в курсе, шт	Время выполнения проверок целостности на уровне приложения, мс	Время выполнения проверок целостности на уровне базы данных, мс
100	24	2
200	12	1
300	13	2
400	24	5
500	37	8
600	47	11
700	52	11
800	27	7
900	48	11
1000	51	11
1100	76	22
1200	107	46
1300	72	21
1400	112	37
1500	100	32
1600	164	60
1700	118	40
1800	287	110
1900	190	71
2000	166	61

По таблице 16 был построен график 7.

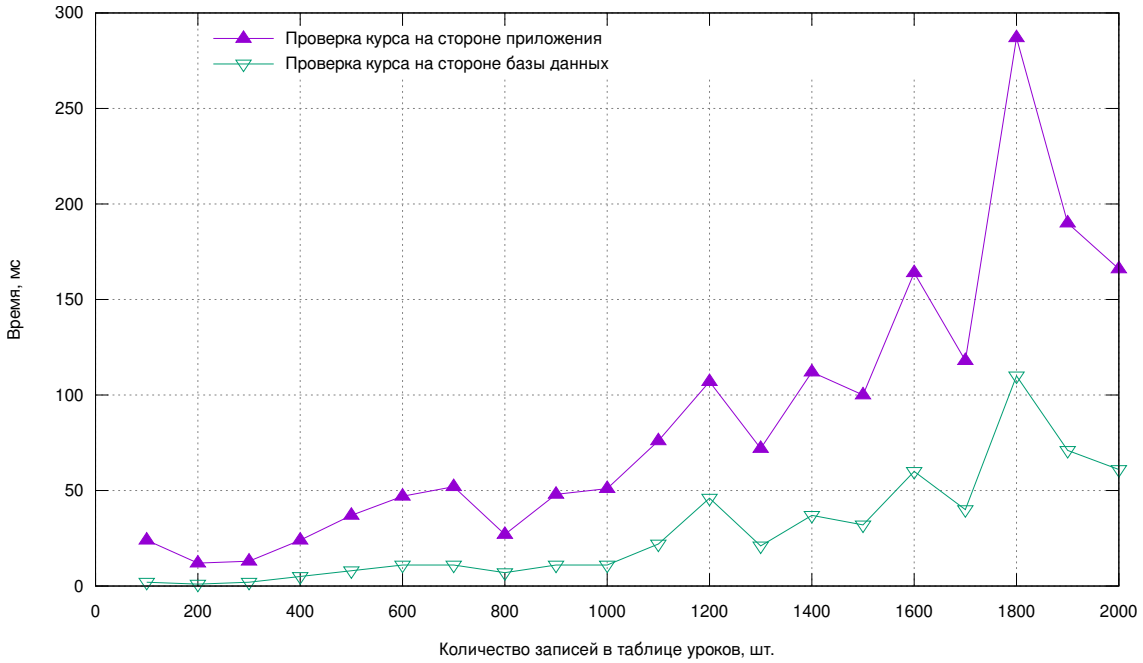

Рисунок 7 – График зависимости времени проверок целостности от количества уроков в курсе при диапазоне значений от 100 до 2000

4.4 Вывод из исследовательской части

По результатом проведенного исследования видно, что проверки целостности курса на стороне приложения в 2-2.5 раза медленнее, чем проверки на уровне базы данных, с помощью триггера. При этом время выполнения проверок зависит линейно от количества записей в таблице уроков, поскольку целостность курса зависит от его наполнения уроками и требованиями, описанными ранее.

Заключение

В ходе выполнения данной курсовой работы были выполнены следующие задачи:

— проанализированы существующие решения;

— спроектирована база данных;

— спроектировано приложение доступа к базе данных;

— реализовано приложение доступа к базе данных;

— исследована зависимость времени проверки целостности данных на уровне приложения и базы данных от количества записей в таблице.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Data Consistency Models: ACID [Электронный ресурс]. — Режим доступа: https://neo4j.com/blog/acid-vs-base-consistency-models-explained (дата обращения: 24.3.2024).
2. MySQL database [Электронный ресурс]. — Режим доступа: https://www.mysql.com/ (дата обращения: 24.3.2024).
3. PostgreSQL documentation [Электронный ресурс]. — Режим доступа: https://www.postgresql.org/docs/ (дата обращения: 24.3.2024).
4. Oracle database [Электронный ресурс]. — Режим доступа: https://www.oracle.com/database/ (дата обращения: 24.3.2024).
5. Try SQL Server on-premises or in the cloud [Электронный ресурс]. — Режим доступа: https://www.microsoft.com/en-us/sql-server/sql-server-downloads (дата обращения: 24.3.2024).
6. Hadoop: Setting up a Single Node Cluster [Электронный ресурс]. — Режим доступа: https://apache.github.io/hadoop/hadoop-project-dist/hadoop-common/SingleCluster.html (дата обращения: 24.3.2024).
7. Open Source NoSQL Database [Электронный ресурс]. — Режим доступа: https://cassandra.apache.org/_/index.html (дата обращения: 24.3.2024).
8. Unified engine for large-scale data analytics [Электронный ресурс]. — Режим доступа: https://spark.apache.org/ (дата обращения: 24.3.2024).
9. What Is SQLite? [Электронный ресурс]. — Режим доступа: https://www.sqlite.org/ (дата обращения: 24.3.2024).
10. Облачное объектное хранилище - Amazon S3 [Электронный ресурс]. — Режим доступа: https://aws.amazon.com/ru/pm/serv-s3/?gclid=CjwKCAjwxY-3BhAuEiwAu7Y6s3wlfKrCd0HOzurPo8AHgPxKA8MHUNa96K05xDqcE2IHwPvcyIICkBwE&trk=b45f363b-5d02-4b3f-87df-b7b1908ff05c&sc_channel=ps&ef_id=CjwKCAjwxY-3BhAuEiwAu7Y6s3wlfKrCd0HOzurPo8AHgPxKA8MHUNa96K05xDqcE2IHwPvcyIICkBwE:G:s&s_kwcid=AL!4422!3!536452769228!e!!g!!amazon%20s3! (дата обращения: 24.3.2024).

12198535626 ! 120978772550 & trk = b45f363b - 5d02 - 4b3f - 87df - b7b1908ff05c & sc _ channel = ps & targetid = kwd - 1745724519 (дата обращения: 24.3.2024).

11. Object storage for companies of all sizes [Электронный ресурс]. — Режим доступа: https://cloud.google.com/storage (дата обращения: 24.3.2024).

12. The MinIO Enterprise Object Store: Built for AI Data Infrastructure [Электронный ресурс]. — Режим доступа: https://min.io/?utm_term=&utm_campaign=&utm_source=adwords&utm_medium=ppc&hsa_acc=8976569894&hsa_cam=20593618271&hsa_grp=&hsa_ad=&hsa_src=x&hsa_tgt=&hsa_kw=&hsa_mt=&hsa_net=adwords&hsa_ver=3&gad_source=1&gclid=CjwKCAjwxY-3BhAuEiwAu7Y6s2d4hsQFY-ksLM9qtHxAMK6ZGlc-rNufICZtAdC9-LVBq-21s_6sMRoC-0AQAvD_BwE (дата обращения: 24.3.2024).

13. Go documentation [Электронный ресурс]. — Режим доступа: https://go.dev/doc/ (дата обращения: 24.3.2024).

14. PL/pgSQL — SQL Procedural Language [Электронный ресурс]. — Режим доступа: https://www.postgresql.org/docs/current/plpgsql.html (дата обращения: 24.3.2024).

15. Develop faster. Run anywhere. [Электронный ресурс]. — Режим доступа: https://www.docker.com/ (дата обращения: 24.3.2024).

16. Golang testing package [Электронный ресурс]. — Режим доступа: https://pkg.go.dev/testing (дата обращения: 24.3.2024).

17. What is Mock Testing? [Электронный ресурс]. — Режим доступа: https://www.radview.com/glossary/what-is-mock-testing/ (дата обращения: 24.3.2024).

18. mockery [Электронный ресурс]. — Режим доступа: https://github.com/vektra/mockery (дата обращения: 24.3.2024).

19. Getting started with Testcontainers for Go [Электронный ресурс]. — Режим доступа: https://testcontainers.com/guides/getting-started-with-testcontainers-for-go/ (дата обращения: 24.3.2024).

20. Apple M1 [Электронный ресурс]. — Режим доступа: https://web.archive.org/web/20201110184757/https://www.apple.com/mac/m1/ (дата обращения: 24.3.2024).

21. Welcome to Faker's documentation! [Электронный ресурс]. — Режим доступа: https://faker.readthedocs.io/en/master/ (дата обращения: 24.3.2024).

ПРИЛОЖЕНИЕ А

Листинг А.1 – Создание таблиц базы данных

```
create table public.user (
    id uuid primary key,
    email varchar(255) unique not null,
    password varchar(255) not null,
    name varchar(255) not null,
    surname varchar(255) not null,
    phone varchar(32),
    city varchar(255),
    avatar_url text
);

create table public.school (
    id uuid primary key,
    name varchar(255) not null,
    description text not null,
    owner_id uuid not null,
    foreign key (owner_id) references public.user(id) on delete
       cascade
);

create type course_status as enum ('draft', 'ready',
   'published');

create table public.course (
    id uuid primary key,
    name varchar(255) not null,
    school_id uuid not null,
    level int not null,
    price bigint not null,
    language varchar(255) not null,
    status course_status not null,
    rating float not null,
    foreign key (school_id) references public.school(id) on
       delete cascade
);

create type lesson_type as enum ('theory', 'video', 'practice');
```

```
create table public.lesson (
    id uuid primary key,
    title varchar(255) not null,
    type lesson_type not null,
    score int not null,
    theory_url text,
    video_url text,
    course_id uuid not null,
    foreign key (course_id) references public.course(id) on
       delete cascade
);

create table public.test (
    id uuid primary key,
    task_url text not null,
    options text not null,
    answer text not null,
    score int not null,
    level int not null,
    lesson_id uuid not null,
    foreign key (lesson_id) references public.lesson(id) on
       delete cascade
);

create table public.review (
    id uuid primary key,
    text text not null,
    course_id uuid not null,
    user_id uuid,
    rating int,
    foreign key (course_id) references public.course(id) on
       delete cascade,
    foreign key (user_id) references public.user(id) on delete
       set null
);

create type certificate_grade as enum ('bronze', 'silver',
   'gold');

create table public.certificate (
```

```
    id uuid primary key,
    name varchar(1024) not null,
    score int not null,
    grade certificate_grade not null,
    created_at timestamp not null,
    user_id uuid not null,
    course_id uuid,
    foreign key (user_id) references public.user(id) on delete
       cascade,
    foreign key (course_id) references public.course(id) on
       delete set null
);

create table lesson_stat (
    id uuid primary key,
    score int not null,
    user_id uuid not null,
    lesson_id uuid not null,
    foreign key (user_id) references public.user(id) on delete
       cascade,
    foreign key (lesson_id) references public.lesson(id) on
       delete cascade
);

create table test_stat (
    id uuid primary key,
    score int not null,
    user_id uuid not null,
    test_id uuid not null,
    foreign key (user_id) references public.user(id) on delete
       cascade,
    foreign key (test_id) references public.test(id) on delete
       cascade
);

create table course_student (
    student_id uuid not null,
    course_id uuid not null,
    primary key (student_id, course_id),
    foreign key (student_id) references public.user(id) on
       delete cascade,
```

```
    foreign key (course_id) references public.course(id) on
       delete cascade
);

create table course_teacher (
    teacher_id uuid not null,
    course_id uuid not null,
    primary key (teacher_id, course_id),
    foreign key (teacher_id) references public.user(id) on
       delete cascade,
    foreign key (course_id) references public.course(id) on
       delete cascade
);

create table school_teacher (
    teacher_id uuid not null,
    school_id uuid not null,
    primary key (teacher_id, school_id),
    foreign key (teacher_id) references public.user(id) on
       delete cascade,
    foreign key (school_id) references public.school(id) on
       delete cascade
);
```

Листинг А.2 – Создание ролей базы данных

```
create role root with
    noinherit
    superuser
    createrole
    login
    password 'root';

create role guest with
    noinherit
    login
    password 'guest';

create role authenticated with
    noinherit
    login
    password 'authenticated';
```

```
grant all on all tables in schema public to root;

grant select on table
    public.school,
    public.course,
    public.school_teacher,
    public.user,
    public.review to guest;
grant insert on table public.user to guest;

grant select on all tables in schema public to authenticated;
grant insert, update, delete on table
    public.review,
    public.user,
    public.course,
    public.school,
    public.certificate,
    public.lesson,
    public.test,
    public.lesson_stat,
    public.test_stat,
    public.course_student,
    public.course_teacher,
    public.school_teacher to authenticated;
```