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, искусственный интеллект и системы управления**»

КАФЕДРА «**Программное обеспечение ЭВМ и информационные технологии**»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Разработка базы данных для хранения и обработки данных ветеринарной клиники»

Студент группы **ИУ7-64Б** _____ **Чепиго Д.С.**

(Подпись, дата) (И.О. Фамилия)

Руководитель курсовой работы _____ **Строганов Ю.В.**

(Подпись, дата) (И.О. Фамилия)

***2023** г.*

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)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ7

_____ Рудаков И. В.

«3» марта 2023 г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине

Базы данных

Студент группы **ИУ7-64Б**

Чепиго Дарья Станиславовна

Тема курсовой работы

Разработка базы данных для хранения и обработки данных ветеринарной клиники

График выполнения работы: 25% к 4 нед., 50% к 8 нед., 75% к 13 нед., 100% к 15 нед.

Задание

Провести анализ предметной области ветеринарных клиник. Сформулировать требования и ограничения к разрабатываемой базе данных и приложению для ветеринарной клиники. Провести анализ существующих решений. Сформулировать описание пользователей проектируемого приложения для доступа к базе данных. Спроектировать архитектуру базы данных и ограничения целостности. Спроектировать ролевую модель на уровне базы данных. Выбрать средства реализации. Реализовать спроектированную БД и необходимый интерфейс для взаимодействия с ней. Исследовать характеристики разработанного программного обеспечения.

Оформление курсовой работы:

Расчетно-пояснительная записка на 25-40 листах формата А4. Презентация на 12-18 слайдах.

Дата выдачи задания «3» марта 2023 г.

Руководитель курсовой работы _____ **Строганов Ю. В.**

Студент _____ **Чепиго Д. С.**

РЕФЕРАТ

Расчетно-пояснительная записка 32 с., 8 рис., 7 табл., 22 ист.

БАЗЫ ДАННЫХ, PostgreSQL, ВЕТЕРИНАРНАЯ КЛИНИКА, АРХИТЕКТУРА ПРИЛОЖЕНИЯ, API, GITLAB CI/CD.

Цель работы – разработать базу данных для ветеринарной клиники.

В процессе работы проанализированы существующих решений и базы данных. Была спроектирована ролевая модель базы данных. Проведено исследование зависимости времени обработки информации от ее объема и распределения вычислений между базой данных и приложением.

В результате спроектировано и реализовано программное обеспечение для ветеринарной клиники.

Содержание

ВВЕДЕНИЕ

Ветеринарная клиника является неотъемлемой составной частью современного общества, обеспечивая качественное лечение и уход за домашними животными. В связи с постоянным ростом количества пациентов и услуг, предоставляемых ветеринарными специалистами, возникает необходимость оптимизации процессов обработки, хранения и анализа данных. Актуальность изучения и разработки такой системы обусловлена не только увеличением объемов информации, но и необходимостью обеспечения безопасности, доступности и целостности данных.

Цель работы – разработать базу данных для ветеринарной клиники.

Чтобы достигнуть поставленной цели, необходимо решить следующие задачи:

— провести анализ существующих решений;
— формализовать задачу и данные;
— спроектировать базу данных и приложение;
— выбрать систему управления базами данных и среду разработки;
— реализовать программное обеспечение;
— исследовать зависимость времени обработки данных от их объема и распределения вычислений между базой данных и приложением.

1 Аналитический раздел

1.1 Анализ существующих решений

По оценкам BusinesStat [1], начиная с 2020 года, основной тенденцией в сфере ветеринарных услуг России являются онлайн-консультации с ветеринарами. Также важно предоставлять клиенту информацию о его питомце и истории посещения клиники в электронном виде. Поэтому при анализе существующих решений были выделены следующие критерии:

— наличие мобильного приложения или сайта, через которое можно осуществлять действия в ветеринарной клинике;
— возможность ознакомиться с врачами и услугами клиники;
— наличие личного кабинета для клиента с информацией о его питомцах;
— возможность просмотра истории приемов.

Последние три критерия подразумевают действие через приложение или сайт, то есть онлайн.

Таблица 1 – Сравнение существующих решений

Существующее решение	Приложение/ сайт	Просмотр услуг	Личный кабинет	История приемов
petstory [2]	+	+	+	-
vet.city [3]	+	+	-	-
vetcare24 [4]	+	+	-	+

Из таблицы 1 видно, что ни одно из существующих решений не удовлетворяет всем критериям.

1.2 Формализация задачи

В рамках курсовой работы необходимо спроектировать и разработать базу данных для ветеринарной клиники, а также приложение, позволяющее с ней работать. Разрабатываемое решение должно удовлетворять всем критериям, вы-

двинутым в предыдущем пункте. В ветеринарной клинике имеется разграничение по ролям: гость, авторизованный клиент, сотрудник клиники и администратор. Подробное описание их возможностей представлено на рисунке 1.

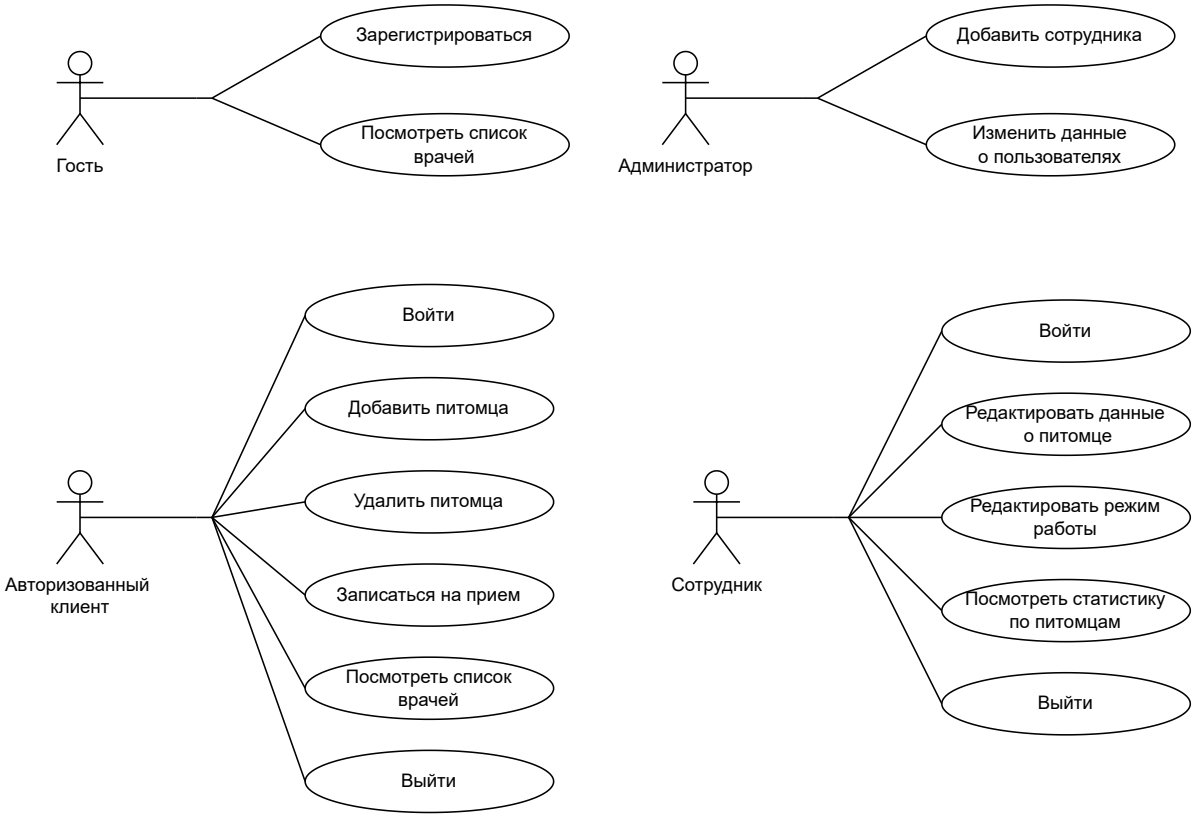

Рисунок 1 – Диаграмма использования приложения

1.3 Формализация данных

База данных должна хранить информацию о:

— работниках клиники;

— клиентах;

— питомцах;

— записях на прием.

ER-диаграмма сущностей в нотации Чена, описывающая сущности, их атрибуты и связи между сущностями в разрабатываемой базе данных, представлена на рисунке 2.

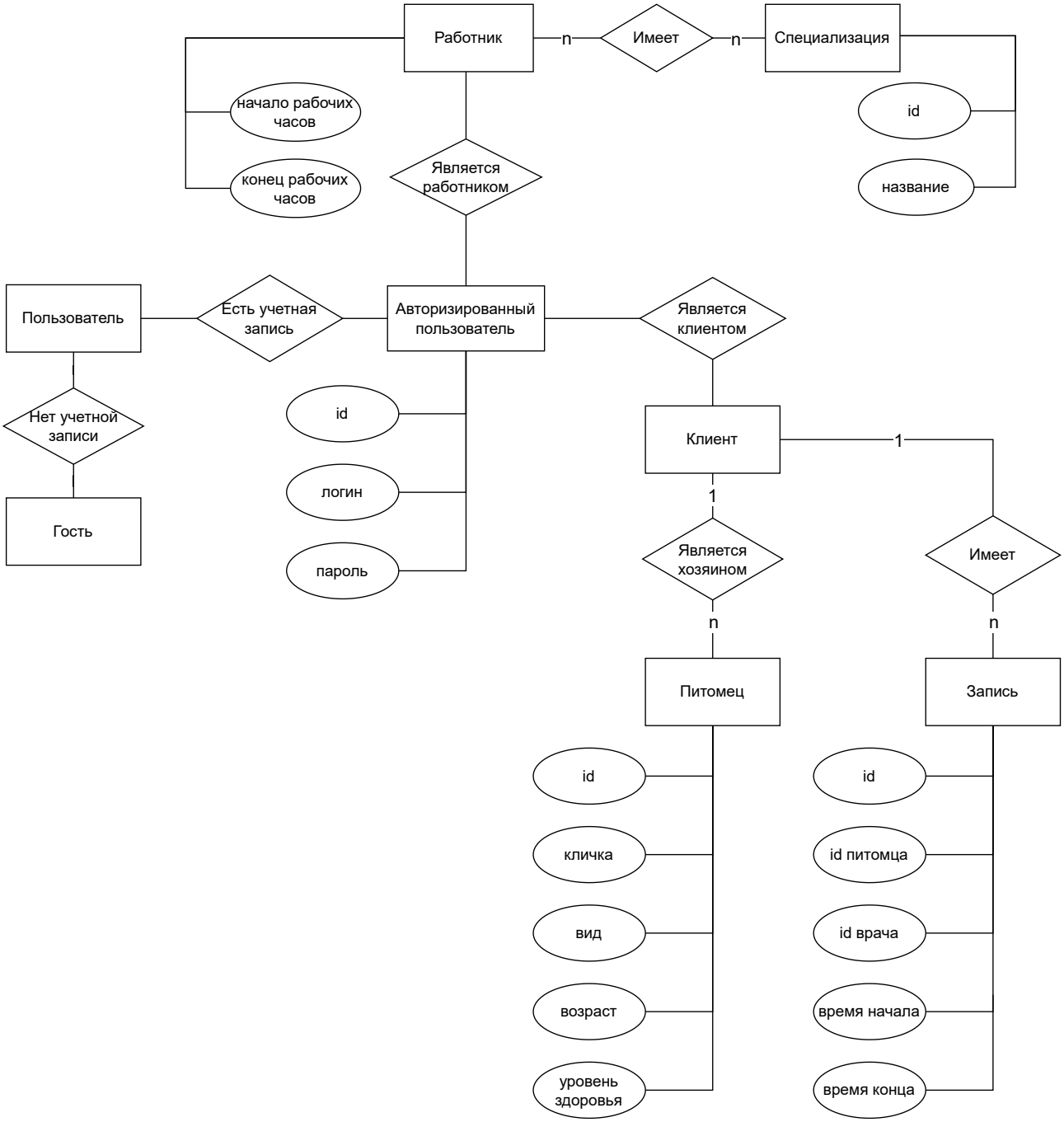

Рисунок 2 – ER-модель в нотации Чена

1.4 Анализ баз данных

По модели хранения базы данных делятся на три группы: дореляционные, реляционные и постреляционные [5].

1.4.1 Дореляционные базы данных

К дореляционным базам данных относятся:

— инвертированные списки;

— иерархичекие;

— сетевые.

База данных на основе инвертированных списков представляет собой совокупность файлов, содержащих записи. Для записей в файле определен некоторый порядок, диктуемый физической организацией данных. Для каждого файла может быть определено произвольное число других упорядочений на основании значений некоторых инвертированных списков.

Иерархическая модель состоит из объектов с указателями от родителей к потомкам, соединяя вместе связанную информацию. Такие базы данных могут быть представлены как дерево.

К основным понятиям сетевой модели относятся: элемент (узел), связь. Узел – это совокупность атрибутов данных, описывающих некоторый объект. Сетевая модель не является полностью независимой от приложения, так как выборка данных зависит от физической организации хранилища. Другими словами, если необходимо изменить структуру данных, то нужно поменять и приложение [6].

1.4.2 Реляционные базы данных

Реляционные базы данных чаще используют язык SQL [7]. Данные реляционных баз хранятся в виде таблиц и строк, таблицы могут иметь связи с другими таблицами через внешние ключи, таким образом образуя отношения [8].

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

Важным свойством реляционных баз данных является их способность удовлетворять требованиям ACID [9]: атомарность, согласованность, изоляция, устойчивость.

1.4.3 Постреляционные базы данных

Постреляционная модель данных является расширенной реляционной моделью, которая снимает ограничение неделимости хранящихся в записях таблиц данных. В таких базах данных реализована работа со сложными типами данных, создаваемых пользователями. Такие решения не всегда могут обеспечить полную поддержку ACID.

Нереляционные хранилища можно использовать вместе с реляционными базами данных. Например, в системах, где основной объем информации хранит SQL, а за кэш отвечает нереляционная база данных [10].

Вывод

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. В качестве системы управления базой данных в данной работе будет использоваться PostgreSQL [11].

2 Конструкторский раздел

2.1 Описание сущностей проектируемой базы данных

В базе данных будет существовать 5 сущностей и 6 таблиц, одна из которых является развязочной.

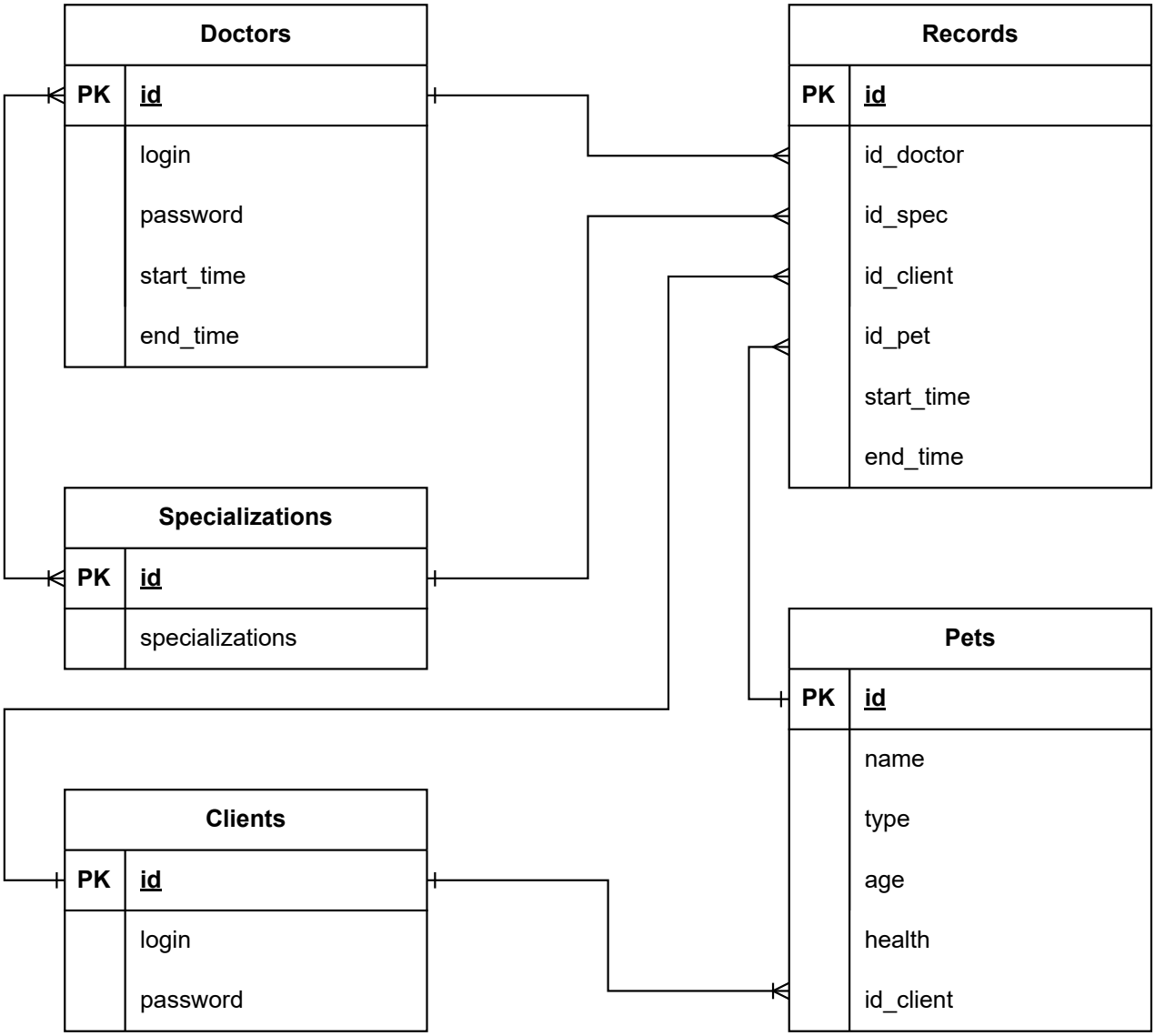

Рисунок 3 – Диаграмма проектируемой базы данных

Необходимы следующие таблицы.

1) Таблица с работниками клиники, то есть докторами (Doctors).
2) Таблица для специализаций работников клиники (Specializations).
3) Развязочная таблица для работников и их специализаций.
4) Таблица с клиентами клиники (Clients).

5) Таблица с питомцами клиники (Pets).
6) Таблица с записями на прием (Records).

В таблице 2 описаны поля таблицы Doctors. В таблице 3 описаны поля таблицы Specializations. В таблице 4 описаны поля таблицы связки для Doctors и Specializations.

Таблица 2 – Поля таблицы Doctors

Поле	Тип данных	Описание
id_doctor	serial	Идентификатор работника
login	text	Идентификатор работника для входа в систему
password	text	Пароль работника для входа в систему
start_time	int	Начало рабочих часов
end_time	int	Конец рабочих часов

Таблица 3 – Поля таблицы Specializations

Поле	Тип данных	Описание
id_spec	serial	Идентификатор специализации
spec_name	text	Название специализации

Таблица 4 – Поля развязочной таблицы Doctors-Specializations

Поле	Тип данных	Описание
id_doctor	int	Идентификатор доктора
id_specialization	int	Идентификатор специализации

В таблице 5 описаны поля таблицы Clients. Клиенты и работники имеют 3 одинаковых поля, но они разнесены по разным таблицам, ибо у них разные

роли в бизнес-процессах. Разделение позволяет лучше структурировать информацию и дополнительные связи с другими сущностями.

Таблица 5 – Поля таблицы Clients

Поле	Тип данных	Описание
id_client	serial	Идентификатор клиента
login	text	Идентификатор клиента для входа в систему
password	text	Пароль клиента для входа в систему

В таблице 6 описаны поля таблицы Pets. В данном проекте у питомца может быть только один владелец, но у владельца может быть много питомцев. Поэтому информация о связи питомец-владелец реализуется с помощью таблицы Pets, где содержится идентификатор хозяина. В таблице 7 описаны поля таблицы Records, которая хранит записи на приемы.

Таблица 6 – Поля таблицы Pets

Поле	Тип данных	Описание
id_pet	serial	Идентификатор питомца
name	text	Кличка питомца
type	text	Вид, порода питомца
age	int	Возраст питомца
health	int	Уровень здоровья питомца
id_client	int	Идентификатор хозяина

Таблица 7 – Поля таблицы Records

Поле	Тип	Описание
id_record	serial	Идентификатор приема
id_doctor	int	Идентификатор доктора
id_spec	int	Идентификатор специализации доктора
id_pet	int	Идентификатор питомца
start_time	int	Время начала приема
end_time	int	Время конца приема

2.2 Ролевая модель

Исходя из сценариев приложения, представленных на рисунке 1 можно выделить следующие роли.

1) Гость. Имеет права только на просмотр таблиц сотрудников и их специализаций.
2) Клиент. Имеет права на просмотр и изменение записей, связанных с ним и его питомцами в таблицах Clients, Pets, Records. Как и гость может смотреть список сотрудников и их специализаций.
3) Работник. Имеет права на просмотр всех таблиц, кроме таблицы Clients. В таблице Clients не может видеть и изменять пароль клиента, то есть поле password. Может изменять данные о себе, о записях на приемы и о питомцах, связанных с ним.
4) Администратор. Имеет права на создание и обновление всех таблиц.

2.3 Триггер базы данных

Для добавления новой записи на прием нужно убедиться, что она валидна. Для этого необходим триггер, реализующий анализ уже существующих записей в таблице для валидации, что желаемое время соответствует часам приема доктора, оно не занято и питомец принадлежит данному клиенту.

Вывод

В данном разделе были описаны сущности, ролевая модель и функции проектируемой базы данных.

3 Технологический раздел

3.1 Архитектура приложения

Для приложения была выбрана клиент-серверная архитектура. Доступ к серверной части будет осуществляться с помощью API [12]. Для осуществления запросов к базе данных будут использоваться коннекторы, предоставляющие интерфейс взаимодействия посредством языка программирования. Приложение было разделено на компоненты: компонент доступа к данным, компонент бизнес-логики, компонент интерфейса. Верхнеуровневое разбиение на компоненты представлено на рисунке 4.

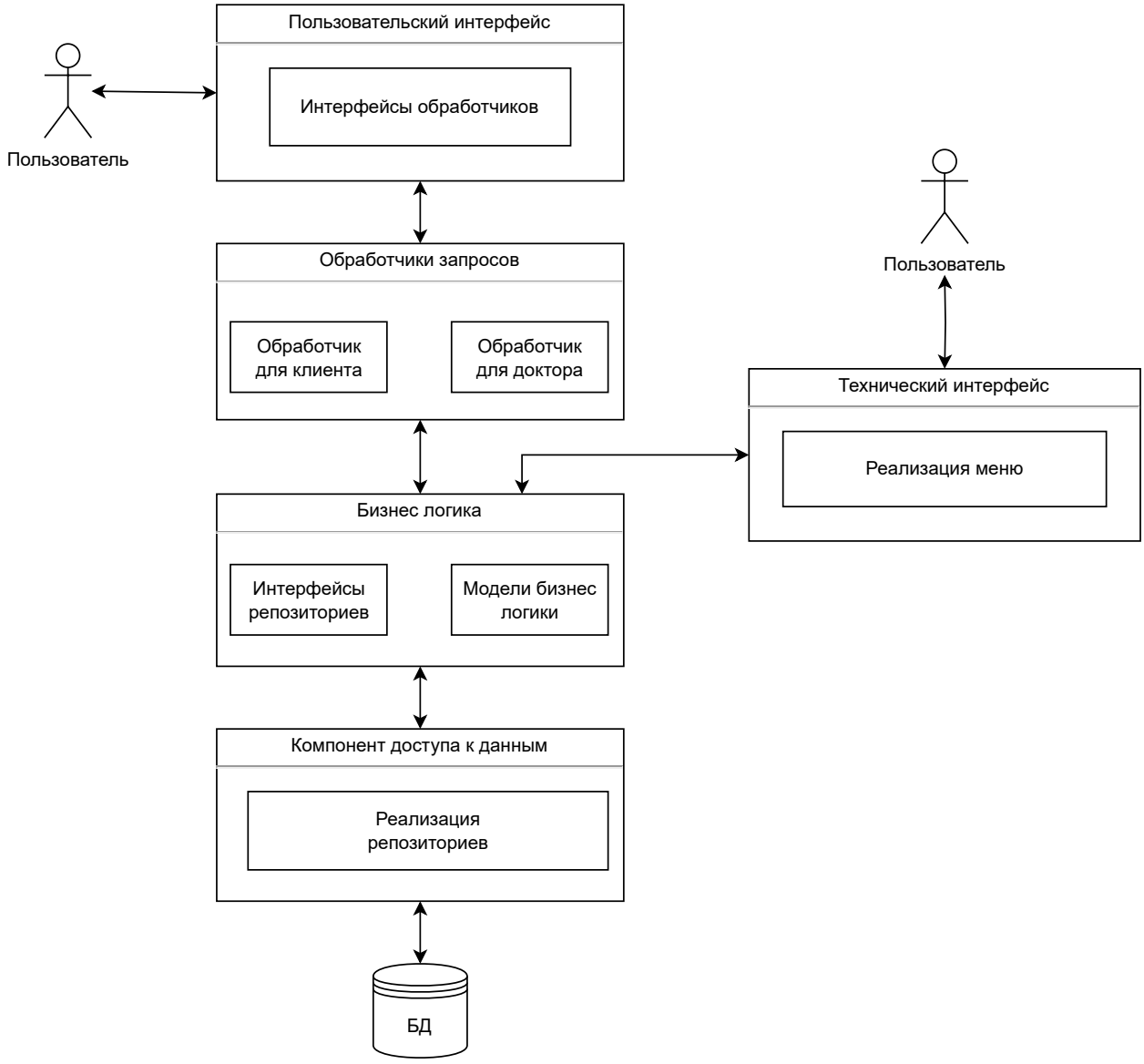

Рисунок 4 – Верхнеуровневое разбиение на компоненты

3.2 Средства реализации

В рамках данной работы были выбраны следующие технологии.

1) Язык программирования – Golang [13].
2) Система управления базами данных – PostgreSQL.
3) Для написания хранимых процедур базы данных будет использовано расширение языка SQL – PL/pgSQL [14].
4) Для работы с СУБД была выбрана библиотека database/sql [15], предоставляющая универсальный интерфейс для реляционных баз данных, а также ее расширение – библиотека sqlx [16].
5) Фреймворк для реализации API – gin [17].
6) Для обечпечения безопасности паролей пользователей была использована хэш-функция bcrypt [18].
7) Аутентификация была реализована на основе Bearer-токенов [19].
8) Для автоматизации развертывания и изолирования приложения была выбрана платформа Docker [20].
9) Технический интерфейс – консоль.

3.3 Детали реализации

3.3.1 Создание таблиц

В листинге 1 представлен код создания таблиц и ограничений, описанных ранее.

Листинг 1: Скрипт создания таблиц

```
drop table if exists doctors cascade;
create table doctors
(
    id_doctor serial primary key,
    login text,
    password text,
    start_time int,
    end_time int
```

```
);
alter table doctors add constraint
    unique_login_doctor unique (login);


drop table if exists clients cascade;
create table clients
(
    id_client serial primary key,
    login text,
    password text
);
alter table clients add constraint
    unique_login_client unique (login);

drop table if exists pets cascade;
create table pets
(
    id_pet serial primary key,
    name text,
    type text,
    age int,
    health int,
    id_client int references clients(id_client)
);

drop table if exists records cascade;
create table records
(
    id_record serial  primary key,
    id_doctor int references doctors (id_doctor),
    id_pet int references pets (id_pet) on delete cascade,
    id_client int references clients (id_client),
    time_start timestamp,
    time_end timestamp
```

```
);

drop table if exists specializations cascade;
create table specializations
(
    id_spec serial primary key,
    spec_name text
);

drop table if exists doctors_specializations cascade;
create table doctors_specializations
(
    id_spec int references specializations(id_spec)
        on delete cascade,
    id_doctor int references doctors(id_doctor)
        on delete cascade,
    primary key (id_spec, id_doctor)
);
```

3.3.2 Создание ролей на уровне базы данных

В конструкторской части были выделены 4 роли на уровне базы данных: гость, клиент, доктор и администратор. Создание ролей и выделение им прав, в соответствии с ролевой моделью, представлены в листинге 2.

Листинг 2: Скрипт создания ролевой модели базы данных

```
create role guest login;
grant select on doctors to guest;

create role client login;
grant select, insert, update(login, password)
    on clients to client;
grant usage, select on sequence clients_id_client_seq to client;
grant select  on specializations to client;
```

```
grant select on doctors_specializations to client;
grant select on doctors to client;
grant select, insert, delete, update on pets to client;
grant usage, select on sequence pets_id_pet_seq to client;
grant select, insert on records to client;
grant usage, select on sequence records_id_record_seq to client;

create role doctor login;
grant select, update(login, password, start_time, end_time)
    on doctors to doctor;
grant usage, select on sequence doctors_id_doctor_seq to doctor;
grant select, insert on specializations to doctor;
grant select, insert on doctors_specializations to doctor;
grant select (id_client, login) on clients to doctor;
grant select, update on pets to doctor;
grant select, insert on records to doctor;

create role administrator login superuser;
```

3.3.3 Создание триггера

В конструкторской части был разработан триггер для проверки валидности новой записи с помощью расширения PL/pgSQL, используемого в СУБД PostgreSQL. Код триггера представлен в листинге 3.

Листинг 3: Триггер проверки валидности новой записи

```
create or replace function check_record()
returns trigger as $$
declare
    overlapping_count integer;
    is_within_working_hours boolean;
begin
    select count(*)
    into overlapping_count
```

```
    from records
    where id_doctor = new.id_doctor
        and time_start = new.time_start;

    if overlapping_count > 0 then
        raise exception
       'a record with the same doctor and start time
 already exists. please choose a different time.';
    end if;


    select (extract(hour from new.time_start) >= d.start_time
        and extract(hour from new.time_start) <= d.end_time)
        and (extract(hour from new.time_end) >= d.start_time
        and extract(hour from new.time_end) <= d.end_time)
    into is_within_working_hours
    from doctors d
    where d.id_doctor = new.id_doctor;

    if not is_within_working_hours then
        raise exception
           'the appointment time is outside the doctor''s
working hours. please choose a correct time.';
    end if;

    return new;
end;
$$ language plpgsql;

create trigger tr_check_record
before insert on records
for each row
execute function check_record();
```

3.4 Тестирование

Для тестирования проекта были реализованы модульные тесты для компонента доступа к данным и для компонента бизнес-логики. Также написаны интеграционные тесты для связи двух компонентов. Каждый тест выполняется в отдельном Docker контейнере, до теста запускается скрипт создания таблиц и в случае необходимости база данных заполняется тестовыми данными.

Для автоматизации тестирования, проведения исследования и развертывания использовался инструмент Gitlab CI/CD [21]. Был создан сценарий, состоящий из четырех стадий.

1) Pre. Стадия состоит из статического анализа кода и инициализации модулей.
2) Test. Стадия содержит модульное тестирование компонента бизнес-логики и компонента доступа к данным, а также интеграционное тестирования.
3) Build. Состоит из сборки серверной части приложения и технического интерфейса.
4) Research. Содержит исследование, описанное в следующем разделе.

Задания, входящие в каждую из стадий приведены на рисунке 5.

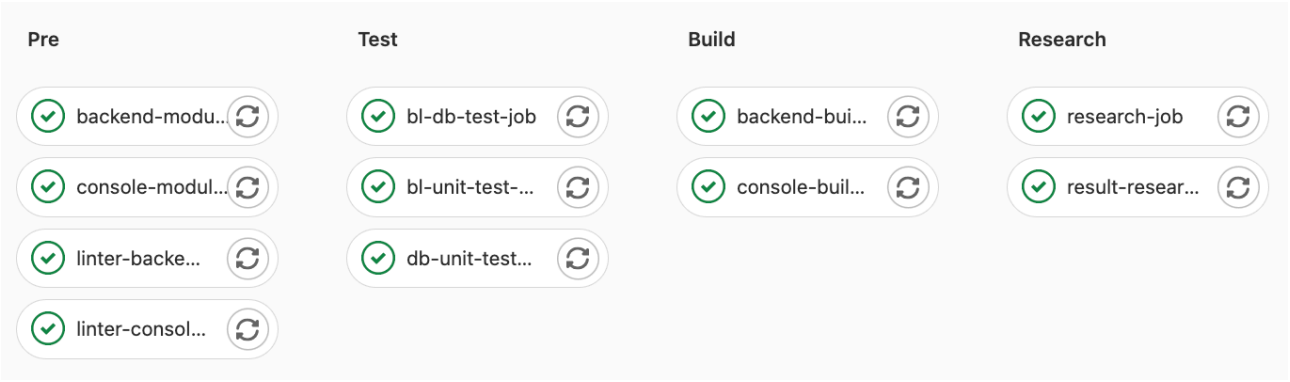

Рисунок 5 – Стадии сборочной линии

Серверная часть приложения и технический интерфейс не связаны между собой, но имеют общие стадии. На рисунке 6 изображена зависимости между заданиями сборочной линии.

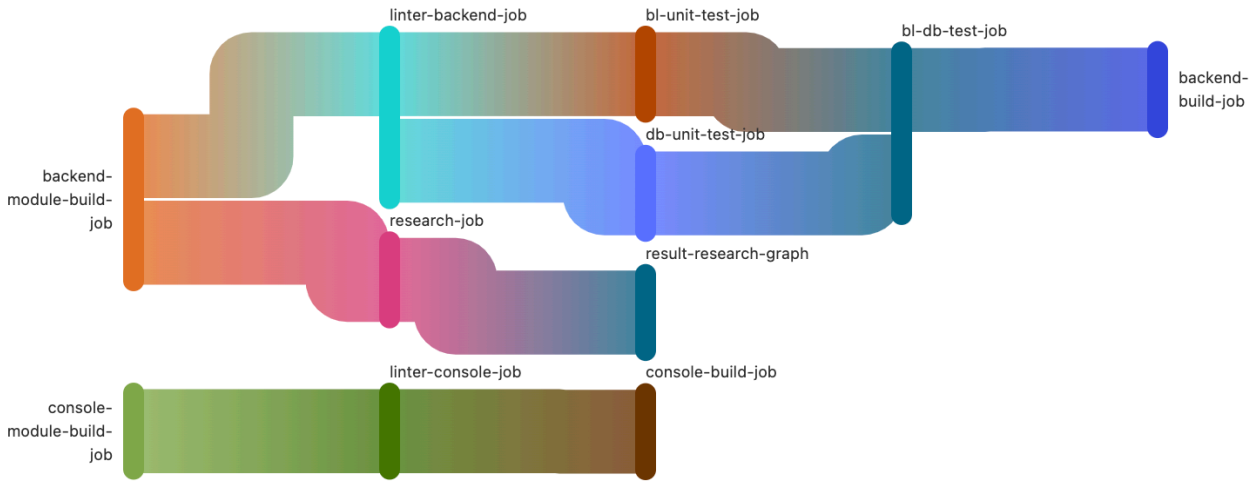

Рисунок 6 – Зависимости между заданиями

Вывод

В данном разделе была описана архитектура приложения, средства и детали его реализации. Приведена сборочная линии для тестирования и развертывания приложения, а также проведения исследования с помощью инструмента Gitlab CI/CD.

4 Исследовательский раздел

4.1 Цель проводимых измерений

Целью является исследование зависимости времени проверки валидности новой записи на прием от общего количества записей и места, где производятся вычисления: на уровне базы данных или приложения.

4.2 Описание исследования

При добавлении новой записи на прием необходимо проверить много факторов: время для записи к врачу попадает в интервал его рабочих часов, питомец принадлежит клиенту, время начало записи не позже времени конца и выбранное время свободно. Эти проверки по-отдельности можно выполнять как на уровне базы данных, так и на уровне приложения. Также часть проверок можно вынести в интерфейс.

Для сравнения времени были выбраны следующие критерии:

— выбранное время для записи свободно;

— время начала и конца приема не противоречит рабочим часам доктора;

Остальные проверки выполняются только на уровне приложения и не учитываются в замерах времени.

Для исследования зависимости времени от объема таблица с записями, количество записей будет последовательно увеличиваться от 10 до 1000. Число сотрудников клиники постоянно и равно 50, их расписание и специализации были сгенерированы случайно. Количество клиентов и питомцев также не меняется со временем и равно 500 и 550 соответственно. Для временного промежутка записей на прием был выбран период в 12 месяцев. Псевдоподобные тестовые данные были сгенерированы с помощью библиотеки Faker [22].

Далее, для добавления новой записи будет случайно выбран доктор, его специализация и время.

4.3 Результаты исследования

Исследования на уровне базы данных и на уровне приложения проводились в разных докер-контейнерах. Перед исследованием база данных заполнялась тестовыми данными, для тестирования на уровне базы данных также создавался триггер.

Так как выбор доктора и время для новой записи выбиралось случайно, то не всегда получалось добавить ее в таблицу. Такие ситуации не были учтены в исследовании. Для каждого размера таблиц случайная новая запись должна была успешно добавиться в таблицу 500 раз. Из-за возникновения ситуации, что запись не может быть добавлена, происходила перегенерация записи.

График зависимости количества перегенераций от размера таблицы с записями представлен на рисунке 7.

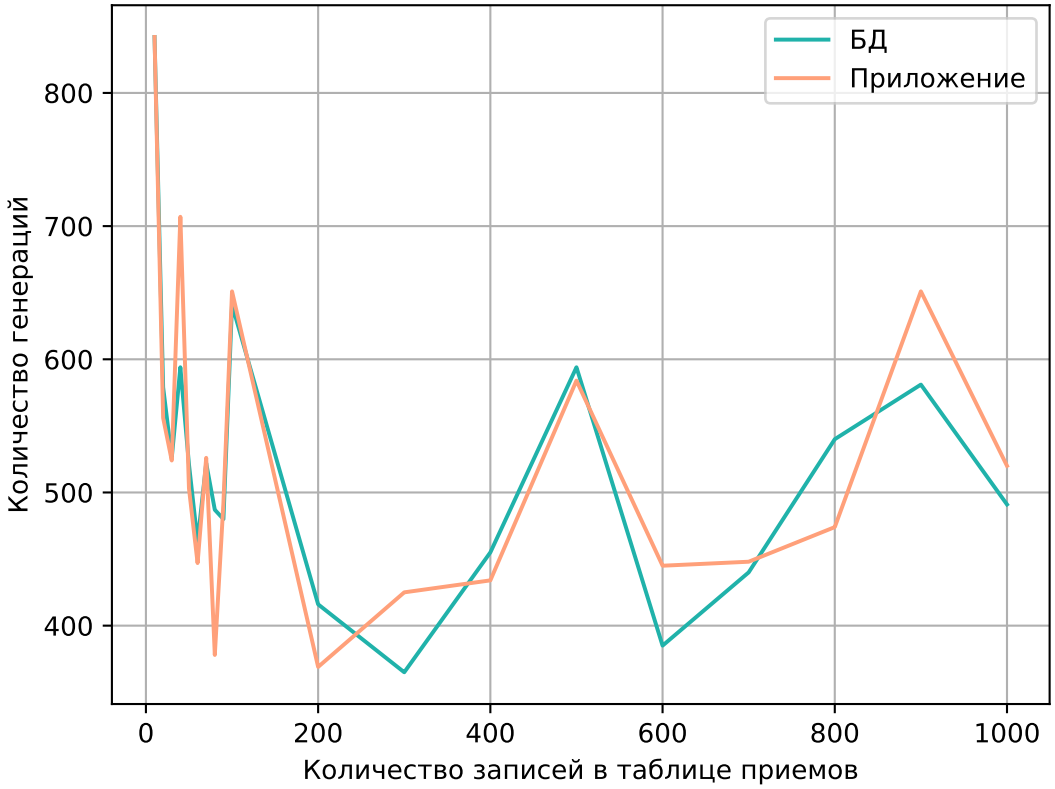

Рисунок 7 – Зависимость количества перегенераций записи на прием от размера таблицы с записями

График зависимости времени проверки новой записи от объема данных и места, где происходят вычисления, представлен на рисунке 8.

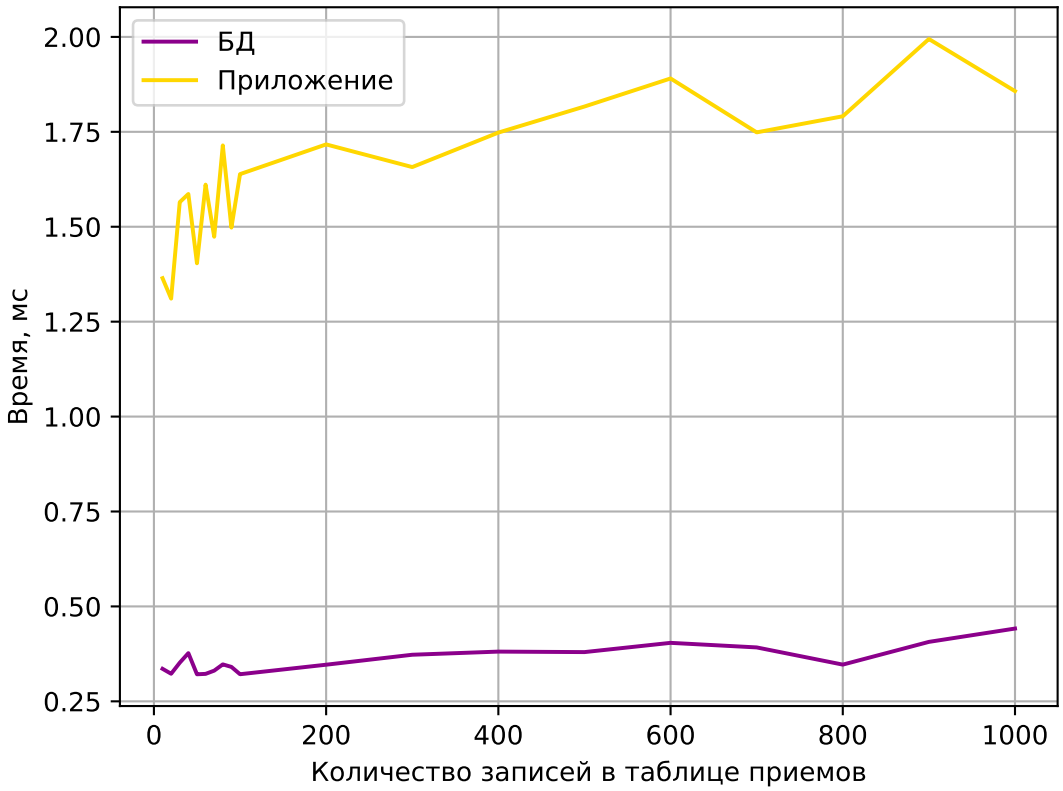

Рисунок 8 – Зависимость времени проверки новой записи на прием от размера таблицы и места, где осуществляются вычисления

По результатам исследования видно, что использование триггера на уровне базы данных в среднем в 5 раз быстрее, чем использование проверок на уровне приложения.

Также на графике присутствуют колебания до того момента, как число записей в таблице не стало больше 100. Это связано с количеством докторов, так как при таком объеме записей каждый доктор имеет 2-3 приема, а временной промежуток для них составляет год.

Далее видно, что время для верификации новой записи на уровне приложения увеличивается быстрее, чем на уровне базы данных. Это связано с объемом передаваемой информации между приложением и базой данных, который

увеличивается с каждой итерацией, так как при обработке на уровне приложения необходимо получить таблицу с записями на прием и таблицу с докторами, которая не меняется на протяжении исследования. Таким образом на уровне приложения выполняется два запроса к базе данных, что и замедляет суммарное время проверки новой записи.

Вывод

В данном разделе было описано исследование времени проверки валидности новой записи на прием от количества записей и места, где осуществляются вычисления. Исследование показало, что эффективне по времени использовать обработку записи на уровне базы данных.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы достигнута поставленная цель: разработана база данных для ветеринарной клиники. В ходе выполнения работы были решены все задачи:

— проведен анализ существующих решений;
— формализована задача и данные;
— спроектирована база данных и приложение;
— выбрана система управления базами данных;
— реализовано программное обеспечение;
— проведено исследование зависимости времени обработки данных от их объема и распределения вычислений между базой данных и приложением.

Разработанную базу данных далее можно совершенствовать, расширяя сценарии использования и добавление кэширования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Маркетинговое агентство BusinesStat, занимающееся исследованием конъюнктуры рынка [Электронный ресурс]. – Режим доступа: https://businesstat.ru/, свободный – (12.03.2023)

2. Ветеринарная клиника «Petstory» [Электронный ресурс]. – Режим доступа: https://petstory.ru/, свободный – (12.03.2023)

3. Ветеринарная клиника «Vetcity Clinic» [Электронный ресурс]. – Режим доступа: https://vet.city/, свободный – (12.03.2023)

4. Ветеринарная клиника «Vetcare24» [Электронный ресурс]. – Режим доступа: https://vetcare24.ru/, свободный – (12.03.2023)

5. К. Дж. Дейт. «Введение в системы баз данных», 8-е издание, издательский дом «Вильяме», 2005. – 1238 с

6. Томас Коннолли, Каролин Бегг. «Базы данных: Проектирование, реализация и сопровождение. Теория и практика», 3-е издание, издательский дом «Вильяме», 2017. – 1440 с

7. What is a Relational Database (RDBMS)? [Электронный ресурс]. – Режим доступа: https://www.oracle.com/database/what-is-a-relational-database/, свободный – (30.05.2023)

8. TECHNICAL SCIENCE / «Colloquium-journal» #2(54),2020, Васильева К.Н., Хусаинова Г.Я, Реляционные базы данных, 2020

9. What is ACID Compliance? [Электронный ресурс]. – Режим доступа: https://www.mongodb.com/databases/acid-compliance, свободный – (12.03.2023)

10. What Is an In-Memory Database? | AWS Amazon [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/ru/nosql/in-memory/, свободный – (12.03.2023)

11. PostgreSQL: The World’s Most Advanced Open Source Relational Database [Электронный ресурс]. – Режим доступа: https://www.postgresql.org/, свободный – (12.03.2023)

12. What is API? [Электронный ресурс]. – Режим доступа: https://www.ibm.com/topics/api свободный – (12.03.2023)

13. Golang – компилируемый многопоточный язык программированияе [Электронный ресурс]. – Режим доступа: https://pkg.go.dev/std/, свободный – (12.03.2023)

14. PL/pgSQL — SQL Procedural Language [Электронный ресурс]. – Режим доступа: https://www.postgresql.org/docs/current/plpgsql.html, свободный – (12.03.2023)

15. Библиотека database/sql [Электронный ресурс]. – Режим доступа: https://pkg.go.dev/database/sql/, свободный – (12.03.2023)

16. Библиотека sqlx [Электронный ресурс]. – Режим доступа: https://pkg.go.dev/github.com/jmoiron/sqlx, свободный – (12.03.2023)

17. Gin Web Framework. The fastest full-featured web framework for Go. [Электронный ресурс]. – Режим доступа: https://gin-gonic.com/, свободный – (12.03.2023)

18. Package bcrypt implements Provos and Mazières’s bcrypt adaptive hashing algorithm. [Электронный ресурс]. – Режим доступа: https://pkg.go.dev/golang.org/x/crypto/bcrypt/, свободный – (12.03.2023)

19. Bearer authentication (also called token authentication) is an HTTP authentication scheme that involves security tokens called bearer tokens. [Электронный ресурс]. – Режим доступа: https://swagger.io/docs/specification/authentication/bearer-authentication/, свободный – (12.03.2023)

20. Docker is a platform designed to help developers build, share, and run modern application. [Электронный ресурс]. – Режим доступа: https://docs.docker.com/, свободный – (12.03.2023)

21. GitLab CI/CD is a tool for software development using the continuous methodologies: [Электронный ресурс]. – Режим доступа: https://docs.gitlab.com/ee/ci, свободный – (30.05.2023)

22. Faker is a Python package that generates fake data for you.: [Электронный ресурс]. – Режим доступа: https://faker.readthedocs.io/en/master/, свободный – (30.05.2023)