МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ВЕБ-ПРИЛОЖЕНИЯ С ЭКСПЕРТНОЙ СИСТЕМОЙ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ JAVA С ИСПОЛЬЗОВАНИЕМ ФРЕЙМВОРКА SPRING BOOT

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Ильичева Михаила Викторовича

Научный руководитель
доцент, к. ф.-м. н _____ А. С. Иванова

Заведующий кафедрой
доцент, к. ф.-м. н _____ А. С. Иванов

Саратов 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире приложения должны быть универсальными. Люди могут пользоваться ими на телефонах с разными операционными системами, а так же пользоваться аналогичными функциями в браузере, просматривая веб-страницу. В каждом, из приведенных выше примеров использования, данные берутся с сервера, машины, на которой запущено приложение, которое обрабатывает все запросы пользователей и выдаёт нужную информацию в ответе.

Цель данной работы — разработать серверное приложение «Happy Cats» с REST api для возможности работы с различными клиентами, используя современные методологии разработки программного обеспечения. Оно позволяет хозяевам кошек узнать больше о породах кошек и заболеваниях, которым подвержена конкретная порода и о всех кошачьих заболеваниях в целом. Данное приложение направлено на то, чтобы владельцы кошек лучше узнали своих питомцев и как следствие стали более ответственными за их жизнь.

Также целью работы является разработать и реализовать экспертную систему для определения болезни кошки.

Выпускная квалификационная работа состоит из введения, 2 разделов, заключения, списка использованных источников и 12 приложений. Общий объем работы — 74 страницы, из них 43 страницы — основное содержание, включая 10 рисунков, список использованных источников информации — 24 наименования.

1 Краткое содержание работы

Первый раздел «Технологии разработки современных приложений» посвящен основной теории по разработке серверного приложения на языке программирования Java с использованием фреймворка Srping Boot. Данный раздел состоит из 9 подразделов.

Первый подраздел «Основные модели процесса разработки» посвящен основным моделям процесса разработки современного приложения.

Методология проектирования програмного обеспечения представляет собой описание того, как нужно организовывать шаги при разработке программного обеспечения, от самых первых шагов, до окончания использования программного обеспечения. Для каждого этапа определяются состав и последовательность выполняемых работ, получаемые результаты, методы и средства, необходимые для выполнения работ, роли и ответственность участников и т.д. Такое формальное описание жизненного цикла информационной системы позволяет спланировать и организовать процесс коллективной разработки и обеспечить управление этим процессом [1] [2].

Список рассмотренных в работе моделей:

— Каскадная модель;
— Инкрементная модель;
— Спиральная модель.

В ходе разработки данного приложения была использована каскадная модель. Следуя всем этапам, изначально было сформировано задание, какое приложение необходимо, затем была спроектирована модель базы данных, обсуждена основная логика работы, а затем основными оказались этапы разработки и тестирования. При использованиях рабочих реализаций системы, тестирования, выявлялись недочеты, которые возвращали разработку на предыдущий этап, чтобы написать новые части программы, решающие данные проблемы. В конце работы, приложение было успешно закончено и введено в эксплуатацию.

Второй подраздел «Паттерны проектирования» посвящен паттернам проектирования, которые используются фреймворк Spring Boot для организации работы его модулей.

При разработке современного приложения встречаются проблемы, требующего одинакового подхода для их решения. Для этого были созданы пат-

терны проектирования — повторяемые архитектуры программного обеспечения.

В данной работе используется фреймворк Spring Boot. В нём используется ряд паттернов:

— Inversion of Control;
— Dependency Injection;
— Singletone;
— MVC;
— Front Controller.

Третий подраздел «Java» посвящен описанию истории языка Java.

Java это язык программирования и, одновременно, платформа вычислений разработанная компанией Sun Microsystems в 1995 г. Она используется при разработке большинства современных сайтов и приложений за счет своей хорошей защищенности, надежности и быстроты работы [7]. Для оптимизации работы в некоторых микропроцессорных системах существует аппаратная поддержка кода на Java.

Четвёртый подраздел «Фреймворки Spring Framework, Spring Boot» посвящен описанию достоинств фреймворков Srping Framework и Spring Boot.

Spring — является одним из популярных фреймворков для разработки Enterprice-приложений, который обеспечивает эффективную модель разработки кода и архитектуры проекта [7].

Основной задачей Spring Framework является предоставление возможности гибкого проектирования, а Spring Boot предназначен для уменьшения длины кода и упрощения разработки веб-приложений [10].

Пятый подраздел «ORM» посвящен технологии, используемой фреймворком Spring Boot.

Spring Data Jpa использует спецификацию Java Persistence API, которая предоставляет возможность сохранять в удобном виде Java-объекты в таблицах реляционных базах данных.

Java Persistence API реализует такую концепцию, как ORM (Object-Relational Mapping) — Объектно-Реляционное-Связывание.

Шестой подраздел «JPA» посвящен технологии JPA, используемой для взаимодействия с моделями фреймворком Spring Boot.

Spring JPA использует аннотации `@Entity`, `@Table`, `@Id`, `@GeneratedValue`,

`@Column`.

На основе моделей генерируются таблицы базы данных со связями, с помощью Java Persistence API [14].

Седьмой подраздел «Project Lombok» посвящен технологии, используемой при разработке, для внедрения новых аннотаций, которые упрощают создание базовых методов классов.

Project Lombok — это библиотека Java, которая автоматически подключается к редактору и инструментам для сборки, добавляя новые аннотации для Java. С их помощью у класса генерируются автоматически геттеры, сеттеры, конструкторы и другие часто используемые методы в классах [15].

Восьмой подраздел «REST» посвящен архитектуре, на которой базируется работа приложения.

Приложение для серверной части было разработано с учетом архитектуры REST (Representational State Transfer — «передача состояния представления»), которая используется для распределенных систем, в том числе для построения интернет-приложений и веб-служб [16] [17].

Девятый подраздел «Экспертная система» посвящен теории об экспертных системах и их основных частях.

Экспертная система — программный комплекс, аккумулирующий знания специалистов в конкретных предметных областях и тиражирующие их эмпирический опыт для консультации менее квалифицированных пользователей [18].

Второй раздел «Разработка серверного приложения Happy Cats» посвящен процессу разработки серверного приложения «Happy Cats» и описанию его функций.

При разработке данного приложения была поставлены требования:

1. Возможность работы с любым клиентом через REST api.
2. Разделение компонентов приложения на слои, ответственные за разную логику работы.
3. Проектирование и создание базы данных для работы с сущностями.
4. Экспертная система, помогающая определить болезнь животного.

Данное приложение состоит из трёх уровней, при которых запрос от пользователя проходит через них все, как и ответ пользователю. Также, для работы этих уровней были созданы модели, сущности, которые соответствуют

необходимым данным для работы приложения, на основе которых генерируются таблицы для базы данных, находящейся в оперативной памяти.

Пришедший от клиента запрос приходит в виде `JSON`, представляющего класс `CreateCatDto`, на класс `CatContoller`. В данном контроллере, с помощью инструментов работы с сессиями пользователей, берётся информация о текущем пользователе, а конкретно `username`. Полученные данные из запроса и информация о пользователе отправляются на следующий слой, класс `CatService`.

В сервисе, проверяется наличие данной породы в базе данных, по её названию, с помощью класса `BreedRepository`. Который посылает сгенерированный запрос базе данных. При успешном ответе базы, происходит его обработка с помощью технологии `ORM`. Если запрос успешен, то в сервисе будет получена сущность породы с данным названием, если нет, то придёт `null`. При `null` будет выброшена ошибка, которая сразу отправится на слой контроллера, где контроллер отошлёт клиенту ответ, что запрос неверен.

Далее проверяется наличие пользователя в базе данных аналогично с проверкой наличия породы. Только используется `username`, отправляясь на слой репозитория, класс `UserRepository`.

При успешно найденных данных, создаётся объект класса `Cat`, только без поля `Id`. Получившийся объект отсылается на уровень репозитория, класс `CatRepository`. Если он успешно добавится в базу данных, то в ответ придёт объект со сгенерированным базой данных `Id`.

При успешно выполненных вышеуказанных действия, полученный объект отсылается обратно на слой контроллера. Контроллер отсылает ответ клиенту, включающий информацию о созданном объекте. В ответе клиенту устанавливается HTTP-статус 200 OK, свидетельствующий об успешном выполнении запроса клиента.

Первый подраздел «Контроллеры» посвящен описанию уровня контроллеров приложения.

Контроллеры — классы, отвечающие за прием запросов от пользователей, и отдачу результата, после обработки запроса, обратно пользователю. Контроллеры построены по архитектуре REST.

Для создания REST API в фреймворке Spring предусмотрена аннотация `@RestController`. Она включает в себя аннотацию `@Controller` и анно-

тацию `@ResponseBody`, отвечающую за тип вид возвращаемого контроллером значения. Это упрощает написание REST API, т.к. значение, возвращаемое методом контроллера будет автоматически упаковано в JSON. Аннотация `@RequestMapping` используется для сопоставления веб-запросов с методами Spring Controller. Были написаны следующие контроллеры:

`AuthenticationController` — используется для регистрации новых пользователей, а также для их входа в систему. В нём `@RequestMapping("/auth")` отвечает за адрес, по которому будут идти запросы к этому контроллеру через url на хосте `http://localhost:8080/auth`. Содержащиеся в нём методы, будут прибавлять свою часть адреса, а также указывать HTTP-метод, с помощью которого нужно обращаться к эндпоинту.

Для входа в систему был реализован метод `signin`, в котором аннотация `@PostMapping("/signin")` говорит о том, что данный метод отвечает за POST-запросы по адресу `http://localhost:8080/auth/signin`.

В нём, в качестве тела запроса, используется объект пересылки данных (data transfer object) `RequestBodyDto`, который содержит только необходимые поля для входа в приложение, такие как пароль и логин пользователя.

При проверке валидности введенных данных используется служебный метод данного контроллера `authentificate`.

В нём логин и пароль передаются в метод класса `AuthenticationManager`, включаемый в библиотеку Spring Security. При ошибке валидации, метод вернёт сообщение о ней.

При успешном входе пользователю вернётся ответ с токеном авторизации вида: `Bearer сгенерированная_последовательность_символов`, необходимым для дальнейшей работы с приложением. Данный токен клиент должен будет присылать в заголовке `Authorization`.

Для регистрации в системе был реализован метод `signup`, данный метод отвечает за POST-запросы по адресу `http://localhost:8080/auth/signup`.

В качестве тела запроса, используется `ReqistrationDto`, содержащий данные о регистрации. Пароль, пришедший в теле сообщения, шифруется перед тем, как попасть на слой сервиса. При успешной регистрации клиенту также отправляется ответ с токеном авторизации.

`UserController` — используется для работы с данными о пользователях. Адрес контроллера — `http://localhost:8080/users`.

`CatController` — используется для работы с данными о моделях `Cat`. Адрес контроллера — `http://localhost:8080/cats`.

Данный контроллер также содержит методы для получения информации о модели, с данными которой он работает. Например, метод `getCatById`.

Также были написаны классы `DiseaseController`, `BreedController` и `NewsController`. В них присутствуют схожие методы для получения всего списка или единицы соответствующей сущности по `id`.

Во втором подразделе «Сервисы» описываются уровни сервисов приложения.

Сервисы — классы, отвечающие за бизнес-логику приложения, как нужно обработать те или иные данные, созданы для того, чтобы отделить бизнес-логику от запросов к базе данных. У сервисов есть методы, отвечающие за разные процессы.

Для смены бизнес-логики сервисы были реализованы в виде интерфейсов, поэтому при смене логики работы сервиса, нужно только поменять имплементацию, использующуюся при создании бинов в Spring.

Были реализованы следующие интерфейсы сервисов, использующиеся в соответствующих контроллерах:

Сервис `UserService` и его имплементация `UserServiceImpl`, отвечающие за логику работы с данными пользователей.

В данном сервисе присутствуют методы, отвечающие за логику добавления пользователей, их поиск, а также метод использующийся не только в контроллере, но и в аутентификации, `getUserById`.

С его помощью проверяется есть ли данный пользователь в базе, а если есть, то получают данные о нём.

Сервис `CatService` и его имплементация `CatServiceImpl`, отвечающие за логику работы с данными о моделях `Cat`.

В данном сервисе присутствуют методы, отвечающие за логику поиска всех объектов модели `Cat`, а также метод использующийся репозиторий пользователей, `addCat`.

Третий подраздел «Репозитории» посвящен уровню репозиториев приложения.

Репозитории — классы, отвечающие за запросы к базе данных. К ним относятся интерфейсы: `BreedRepository`, `CatRepository`, `DiseaseRepository`,

NewsRepository, UserRepository. Эти интерфейсы наследуются от интерфейса JpaRepository, который включен в библиотеку Spring Data.

Spring Data — механизм для взаимодействия с сущностями базы данных, организации их в репозитории, извлечение данных, изменение, в каких то случаях для этого будет достаточно объявить интерфейс и метод в нем, без имплементации.

При наследовании от JpaRepository, нужно конфигурировать репозиторий, передавая в качестве параметров с какой сущностью T он будет работать и сигнатуру первичного ключа (Primary Key) данной сущности.

Иногда ненужно объявлять и сами методы, т.к. в этом интерфейсе включены стандартные методы создания, вставки, удаления и обновления сущностей из базы данных, а также поиск по Id, или поиск всех сущностей, т.е. метод реализующий запрос в базу `select * from table_name`.

Т.е., при наследовании от JpaRepository, репозитории имеют следующие методы:

- `long count();`
- `Iterable<V> saveAll(Iterable<V> collection);`
- `Optional<V> findById(Id);`
- `boolean existsById(Id);`
- `Iterable<V> findAll();`
- `void deleteAll(Iterable<V> collection);`
- `void deleteById(Id);`
- `void delete(V entity);`
- `void deleteAll();`
- `V save(V entity);`
- `Iterable<V> findAllById(Iterable<ID> Ids);`

У Srping JPA есть возможность генерации запросов по именам написанных пользователем функций. Как результат этих функций, их сигнатура, может быть получены сама сущность V, Optional<V>, когда может и отдаться null, а также коллекции, содержащие сущности V. Сами имена функций должны состоять из ключевых слов, предоставляемых SpringJPA, таких как: findVBy,countVBy, getVBy, queryVBy, и readVBy, где V — имя самой сущности. Также есть возможность объединение нескольких ключевых слов, условий запросов, при помощи ключевых слов Or,And.

В репозитории `UserRepository` создано несколько составных методов, с настроенными результатами отдачи.

Для поиска в базе пользователя по его логину был сгенерирован метод `findUserByUsername`, использующийся не только в бизнес-логике приложения, но и в методах аутентификации, который принимает логин пользователя, затем, средствами Spring Data Jpa генерируется запрос к базе данных. После выполнения запроса, данный метод возвращает `Optional<User>`. `Optional` в Java нужен для более удобной обработки null значений. Т.е. в данном методе может прийти как и пользователь с данным логином, так и null, если такого пользователя нет. Данный класс обрабатывается в сервисах, перед тем, как отдать результат в контроллер [21].

Четвёртый подраздел «Модели» посвящен уровню моделей приложения.

Для работы с данными создаются модели, которые будут использоваться в приложении. Представлений самих моделей может быть множество, они отвечают за состояния, изменение данных. Реализация представляет собой простые Java-классы. Были созданы модели:

— `User` — модель, отвечающая за пользователя.
— `Cat` — модель, отвечающая за кошек.
— `Disease` — модель, отвечающая за болезни, присущие породам.
— `Breed` — модель, отвечающая за породы кошек.
— `News` — модель, содержащая в себе новостные статьи.

Пятый подраздел «База данных» посвящен описанию структуры базы данных и используемых технологиях для её создания.

Для корректной работы приложения необходимо хранить данные о моделях, их связях и состояниях. Чтобы это осуществить хранение нужно использовать реляционную базу данных и СУБД для неё. Для этого была выбрана СУБД H2.

H2 — открытая кроссплатформенная СУБД, полностью написанная на языке Java. Она может быть встроена в приложения Java или работать в режиме клиент-сервер. База данных H2 может быть настроена для работы как резидентная база данных, т.е. данные будут храниться в оперативной памяти и не будут сохраняться на диске, также можно выбрать режим хранения на диске. Для удобства работы H2 имеет браузерную консоль управления [22].

Так как база данных находится в оперативной памяти, то при каждом старте приложения нужно загружать в неё данные. H2 представляет возможность загружать данные в базу при помощи класса в приложении, выполняя методы из классов для взаимодействия с базой данных, называемыми репозитории, в честь основной своей функции — взаимодействие с хранилищем данных.

В данном приложении используется метод загрузки данных через специальный класс, подготавливающий начальные данные для тестировщиков, администраторов, некоторых пользователей, чтобы можно было использовать все функции, минуя этапы регистрации и создания сущностей, таких как `Cat`, `Disease`, `Breed`, `News`.

В них аннотация `@Component` используется, для того, чтобы Spring внедрил необходимые зависимости, которые отмечены аннотациями `@Autowired`. При вызове таких классов используются уже созданные объекты, находящиеся в памяти, таким образом реализуется принцип внедрения зависимостей [23].

Шестой подраздел «Конфигурация» посвящен описанию настройки приложения.

Для запуска приложения, а также для настройки параметров Spring, был написан файл конфигурации `application.properties` [24].

В нём ряд настроеки отвечают за подключение к базе в оперативной памяти.

В `spring.datasource.url` присвоен адрес созданной в памяти базы данных. Также через этот параметр можно подключить и другую базу данных, например поднятую на другом сервере.

Для внедрения зависимостей и использования конкретных имплементаций сервисов используется специальный класс конфигурации `ImplConfig`.

В нём представлены внедрения зависимостей для классов сервисов. С их помощью, при создании бина будет создаваться конкретная имплементация интерфейса сервиса.

Седьмой подраздел «Экспертная система» посвящен описанию экспертной системы приложения «HappyCats».

Для выявления болезней у кошек, в помощь хозяевам, была разработана экспертная система «HappyCats». С её помощью, ведя осознанный диалог,

можно распознать, что на данный момент с животным. При ответе на вопрос, от клиента отсылается информация система, а она решает, что отослать клиенту дальше: ответ или следующий вопрос.

Система имеет свои три слоя обработки информации. В контроллере `ExpertSystemController` присутствует три метода. При их подключении к интерфейсу, пользователи могут вести диалог с системой, а эксперты добавлять новые вопросы, ответы, состояния.

Для добавления новых вопросов экспертами используется `addquestion` метод.

Для добавления новых ответов, конечных результатов, используется метод `addAnswer`.

Основным методом для диалога с пользователем является `ask`.

Принцип работы заключается в следующем: на клиенте создаётся запрос, который отправляется по адресу `expert/ask/{useranswer}`. В этом запросе, как параметр пути `{useranswer}`, передаются собранные ответы пользователя. Для начала диалога в эту переменную нужно подставить `start`.

Все параметры пути являются ключами в базе данных, для вопросов и ответов. Если ввести несуществующий ключ, то клиенту придёт сообщение о некорректности данных.

Сущности ответов и вопросов содержат в себе поле `value`, из которого клиент берёт текст ответа и вопроса, соотвественно. Так как вопрос не является конечным результатом работы, то у него есть поле `newId`, которое является основой новой переменной для следующего запроса клиента. При ответе на вопрос, клиент берёт строку из поля `newId`, добавляет к ней `y`, при утвердительном ответе, или `n`, при отрицательном ответе, соответственно. Новую полученную строку нужно использовать как параметр пути `{useranswer}`.

Процесс повторяется до тех пор, пока не будет дан ответ.

Изначально, в приложении, были инициализированы данные и для экспертной системы, и для основных сущностей, в классе `DemoData`.

ЗАКЛЮЧЕНИЕ

В ходе написания данной работы был получен опыт разработки серверных приложений с применением технологии фреймворк Spring Boot.

Успешно реализована REST-архитектура приложения, включающая в себя три слоя для обработки запросов пользователей.

Продемонстрированы методы работы приложения «HappyCats», с успешно отправленными запросами и полученными ответами.

Разработа и реализована экспертная система для определения болезни кошки.

Разработана и реализована базы данных для приложения с использованием СУБД H2, с последующим добавлением данных в неё.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 НОУ ИНТУИТ | Лекция | Жизненный цикл программного обеспечения ИС. [Электронный ресурс]. — URL: https://www.intuit.ru/studies/professional_skill_improvements/1901/courses/55/lecture/1620?page=1/ (Дата обращения 04.05.2020). Загл. с экр. Яз. рус.

2 *Гаврилова, И.* Разработка приложений / И. Гаврилова. — Флинта, 2017.

3 *Royce, W.* Managing the development of large software systems // Technical Papers of Western Electronic Show and Convention (WesCon). — Los Angeles, USA: 1970.

4 Модели и методологии разработки ПО | GeekBrains - образовательный портал. [Электронный ресурс]. — URL: https://geekbrains.ru/posts/methodologies (Дата обращения 04.05.2020). Загл. с экр. Яз. рус.

5 Методологии разработки: Waterfall | GeekBrains - образовательный портал. [Электронный ресурс]. — URL: https://geekbrains.ru/posts/waterfall (Дата обращения 04.05.2020). Загл. с экр. Яз. рус.

6 *Швец, А.* Погружение в паттерны проектирования / А. Швец. — Refactoring.Guru, 2018.

7 Java Platform, Standard Edition (Java SE) 8 documentation[Электронный ресурс]. — URL: https://www.java.com/ru/about/whatis_java.jsp?bucket_value=desktop-chrome83-osx10155&in_query=no (Дата обращения 10.05.2020). Загл. с экр. Яз. рус.

8 *Блох, Дж.* Java: эффективное программирование / Дж. Блох. — Издательский дом «Вильямс», 2018.

9 Spring documentation. [Электронный ресурс]. — URL: https://www.spring.io/ (Дата обращения 10.05.2020). Загл. с экр. Яз. англ.

10 *Крейг, У.* Spring в действии / У. Крейг. — ДМК Пресс, 2015.

11 Java, Spring and Web Development tutorials. [Электронный ресурс]. — URL: https://www.baeldung.com/ (Дата обращения 10.05.2020). Загл. с экр. Яз. англ.

12 *Ноубл, Дж.* Flex 4. Рецепты программирования / Дж. Ноубл, Т. Андерсон, Г. Брэйтуэйт, М. Казарио, Р. Третола. — Санкт-Петербург: БХВ-Петербург, 2011.

13 Generate Database Schema with Spring Data JPA | Baeldung [Электронный ресурс]. — URL: https://www.baeldung.com/spring-data-jpa-generate-db-schema (Дата обращения 12.05.2020). Загл. с экр. Яз. англ.

14 Java Persistence API FAQ [Электронный ресурс]. — URL: https://www.oracle.com/technetwork/java/javaee/persistence-jsp-136066.html (Дата обращения 14.05.2020). Загл. с экр. Яз. англ.

15 Project Lombok. [Электронный ресурс]. — URL: https://projectlombok.org/ (Дата обращения 10.05.2020). Загл. с экр. Яз. рус.

16 *Wilde, E.* REST: From Research to Practice / E. Wilde, C. Pautasso. — Springer, 2014.

17 REST — Национальная библиотека им. Н. Э. Баумана[Электронный ресурс]. — URL: https://ru.bmstu.wiki/index.php?title=REST&mobileaction=toggle_view_desktop (Дата обращения 14.05.2020). Загл. с экр. Яз. рус.

18 *Когаловский, М. Р.* Перспективные технологии информационных систем. / М. Р. Когаловский. — Москва: ДМК Пресс; Компания АйТи, 2003.

19 *Джарратано, Д.* Экспертные системы. Принципы разработки и программирования / Д. Джарратано, Г. Райли. — Москва: Издательский дом «Вильямс», 2007.

20 2. JPA Repositories [Электронный ресурс]. — URL: https://docs.spring.io/spring-data/jpa/docs/1.5.0.RELEASE/reference/html/jpa.repositories.html (Дата обращения 10.05.2020). Загл. с экр. Яз. англ.

21 Optional (Java Platform SE 8) [Электронный ресурс]. — URL: https://docs.oracle.com/javase/8/docs/api/java/util/Optional.html (Дата обращения 15.05.2020). Загл. с экр. Яз. рус.

22 H2 tutorial. [Электронный ресурс]. — URL: https://www.h2database.com/html/tutorial.html (Дата обращения 14.05.2020). Загл. с экр. Яз. англ.

23 Spring Bean Annotations | Baeldung [Электронный ресурс]. — URL: https://www.baeldung.com/spring-bean-annotations (Дата обращения 15.05.2020). Загл. с экр. Яз. англ.

24 Common Application properties [Электронный ресурс]. — URL: https://docs.spring.io/spring-boot/docs/current/reference/html/appendix-application-properties.html (Дата обращения 17.05.2020). Загл. с экр. Яз. англ.