Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

РАЗРАБОТКА ОТКАЗОУСТОЙЧИВОГО САЙТА ДЛЯ ХУДОЖНИКОВ С ТРЁХМЕРНОЙ ГАЛЕРЕЕЙ И ПОДДЕРЖКОЙ ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Давыдова Виктора Вениаминовича

Научный руководитель
доцент, к. ф.-м. н. _____ И. А. Батраева

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные стандарты деятельности предприятий требуют высокой мобильности, доступности, кросс-платформенности для приложений, которые обеспечивают их работу. Кроме того во многих случаях предпочтительно снижение нагрузки на клиентов, как с точки зрения обеспечения безопасности, так и с точки зрения будущих обновлений и масштабирования приложения. Поэтому для многих программных продуктов использование веб-технологий при разработке стало насущной необходимостью.

Целью дипломной работы является изучение современных технологий построения веб-сервисов. Для выполнения заданной цели были поставлены следующие задачи:

- изучение функционирования контейнеров сервлетов;
- разработка приложений при помощи фреймворка Spring;
- изучение принципа разработки программного обеспечения IoC (Inversion of Control) — инвертирование управления приложением;
- изучение процесса DI (Dependency Injection) — разрешение (обеспечение), внедрение зависимостей;
- изучение шаблона проектирования DAO (Data Access Object) — объект доступа к данным;
- изучение шаблона проектирования MVC (Model, View, Controller) — модель, представление, контроллер;
- понимание пяти принципов построения программных продуктов SOLID
- запуск сервиса-приложения при помощи docker-swarm, тем самым обеспечивая его отказоустойчивость;
- изучение методов построения трёхмерных сцен и моделей при помощи Three.js.

1 Технологии разработки приложений на основе веб-сервисов

1.1 Контейнер Сервлетов

Веб-сервер — комплекс программ, обеспечивающий выдачу статических веб-страниц пользователям в ответ на их запросы, посредством (HTTP). Он может располагаться, как на физически существующей машине, так и в виртуальной среде.

Контейнер сервлетов (или веб-контейнер) — основная часть веб-сервера, которая отвечает за обработку запросов и выдачу соответствующих им ответов.

Сервлет — интерфейс определённый в пакете `javax.servlet`. Он определяет три основных метода, отвечающих за жизненный цикл сервлета — `init()` (создание и инициализация), `service()` (обработка входящих запросов) и `destroy` (уничтожение, освобождение ресурсов). Они реализуются каждым сервлетом и каждый из них запускается сервером в определённое время.

Исходя из жизненного цикла объекта сервлета можно заключить, что классы сервлетов загружаются в контейнер, загрузчиком классов, динамически. Каждый запрос находится на обработке в собственном потоке, а объект сервлета может одновременно обслуживать несколько потоков. Когда поток более не используется, он должен быть зачищен сборщиком мусора JVM.

Реализация Сервлета является Java-программой и, как и любая программа на Java, сервлет работает в JVM. Чтобы справиться со сложностью HTTP-запросов, появляется контейнер сервлета. Контейнер сервлета отвечает за создание, выполнение и уничтожение сервлетов.

1.2 Принцип разработки ПО IoC

IoC (Inversion of Control) — это принцип разработки программного обеспечения, при помощи которого управление объектами или частями программы передаётся контейнеру или фреймворку. Этот принцип чаще всего используется в контексте объектно-ориентированного программирования.

1.3 Шаблон проектирования DI

DI (Dependency Injection, инъекция зависимостей) — это шаблон проектирования программного обеспечения, при котором один объект (или статический метод) предоставляет зависимости другого объекта. Зависимость —

это объект, который может быть использован (сервис). Инъекция — это передача зависимости зависимому объекту (клиенту), который будет его использовать. Сервис включается в состояние клиента. Передача сервиса клиенту, а не предоставление клиенту возможности создавать или находить сервис, является основополагающим требованием шаблона.

1.4 DAO

В компьютерном программном обеспечении объект доступа к данным (DAO) является объектом, который предоставляет абстрактный интерфейс для какого-либо типа базы данных или другого механизма сохранения данных. Путем сопоставления вызовов приложений на уровне персистентности DAO предоставляет необходимые операции с данными, не раскрывая детали реализации базы данных. Эта изоляция соответствует принципу единой ответственности. Она разделяет доступ к базе данных необходимый приложению с точки зрения объектов и типов данных (публичного интерфейса DAO), от того, как этот доступ может быт выполнен конкретной СУБД, схемой базы данных и т.д. (Реализация DAO).

1.5 DTO

Data Transfer Object (DTO) — объект передачи данных, который осуществляет передачу данных между процессами. Причины для его использования заключается в том, что связь между процессами обычно осуществляется с использованием удаленных интерфейсов (например, веб-служб), где каждый вызов является дорогостоящей операцией. Поскольку большая часть стоимости каждого вызова связана с временем прохода между клиентом и сервером, одним из способов уменьшения количества вызовов является использование объекта (DTO), который объединяет данные, которые были бы переданы несколькими вызовами, в один большой вызов [6].

1.6 Шаблон проектирования MVC

MVC — аббревиатура — Model, View, Controller — модель, представление и контроллер. MVC — подход организации программного кода являющийся хорошей практикой на данный момент. Главная идея MVC заключается в том, что код приложения разделяется по различным ролям. Одни блоки кода отвечают за содержание и управление данными приложения, другие за орга-

низацию внешнего, пользовательского представления приложения, а третьи за обработку запросов поступающих приложению от пользователей [8].

1.7 ORM

ORM (Object-Relational Mapping) — объектно-реляционное отображение, технология программирования, отвечающая за связывание баз данных с тезисами объектно-ориентированных языков программирования. Данная технология отвечает за создание виртуальной, объектной базы данных.

Данная технология решает задачу обеспечения работы с базами данных в терминологии классов-структур, а не таблиц данных. Она отвечает за преобразование моделей классов в языке программирования в структуры, которые можно хранить в базах данных. Данная технологии предоставляет интерфейс для выполнения базовых (CRUD) операций в базе данных вне зависимости от её диалекта.

Hibernate — библиотека Java с открытым исходным кодом реализующая принцип ORM. Фреймворк Hibernate позволяет сопоставлять классы Java с таблицами базы данных и типами данных языка Java с типами данных SQL и предоставляет механизмы для осуществления запросов к базе данных.

1.8 SOLID

Пять принципов проектирования в ООП, предназначенных для того, чтобы делать программные проекты более понятными, гибкими и поддерживаемыми. Такие методологии, как гибкая разработка (Agile) или адаптированная разработка (Adaptive Software) программного обеспечения формируются на основе этих принципов, применимых к любому объектно-ориентированному дизайну. SOLID — термин, описывающий набор принципов разработки для эффективного кода.

Акроним SOLID значит:

- Принцип единственной ответственности (Single-Responsibility Principle, SRP) гласит, что на каждый класс должна быть возложена одна-единственная обязанность. Для изменения класса может существовать только одна причина.
- Принцип открытости/закрытости (Open/Closed Principle, OCP): Сущности (классы, модули, функции и т.п.) должны быть открытыми для расширения, но закрытыми для модификации.

— Принцип подстановки Барбары Лисков (Liskov Substitution Principle, LSP): «объекты в программе должны быть заменяемыми на экземпляры их подтипов без изменения правильности выполнения программы». Наследующий класс должен дополнять, а не изменять базовый.
— Принцип разделения интерфейса (Interface Segregation Principle или ISP): «множество специализированных интерфейсов, лучше, чем один интерфейс общего назначения». Идея состоит в том, чтобы сохранить компоненты ориентированными на решение своих задач и попытаться минимизировать зависимости между ними.
— Принцип инверсии зависимостей (Dependency Inversion Principle или DIP) «Зависимость на Абстракциях. Нет зависимости на что-то конкретное» Зависимости внутри системы строятся на основе абстракций. Модули верхнего уровня не зависят от модулей нижнего уровня.

1.9 Docker

Docker — это мировой лидер программного обеспечения контейнерной платформы. Разработчики используют Docker чтобы исключить проблемы связанные с работой на своём компьютере, когда работаешь совместно с другими людьми. Операторы используют Docker чтобы запускать и управлять приложениями рядом друг с другом в изолированных контейнерах, чтобы получить наилучшую компактность. Во время работы над большими проектами, Docker используется для быстрого выпуска очередной версии продукта чтобы задействовать новые возможности быстрее, более безопасно, как на Linux, так и Windows системах.

1.10 Three.js

Three.js — это кросс-браузерная библиотека JavaScript и интерфейс прикладного программирования (API), используемый для создания и отображения анимированной трёхмерной компьютерной графики в веб-браузере. Three.js использует WebGL. Её исходный код размещен в репозитории на сайте GitHub.

2 Разработка приложения «Хранитель Творческих Миров»

2.1 Проект — Хранитель Творческих Миров

Данный проект создавался с целью изучения современных подходов к построению веб-сервисов, изучения технологий виртуальной реальности, а также для того, чтобы понять, как осуществляется разработка программных продуктов в области знаний никак не связанной с программированием или иными техническими специальностями.

Идея сервиса-сайта заключается в предоставлении художнику платформы, где он сможет сформировать виртуальный мир, в котором должны существовать его произведения искусства. У него должны быть возможности по загрузке своих картин в специальную галерею картин, из которой он их, впоследствии, сможет взять и разместить в трёхмерном, виртуальном мире-галерее. Этот мир должен поддаваться гибкой настройке, к примеру должна быть возможность менять структуру галереи-мира, фон пола, стен и потолка.

У всякого проекта должна быть идея, которая бы вселяла в пользователей интерес к проекту, и желание стать его частью. Так и у данного сервиса есть идея.

Рисунок 1 – Хранитель творческих миров

Она заключается в существовании мифического хранителя творческих миров — мудрого старца-искусствоведа, который может перемещать картины и прочие изделия искусства из нашего мира в тот, что им подходит более всего.

2.1.1 О приложении

Проект написан с использованием технологий Spring MVC и Web Servlets, на языке программирования Java. Веб-приложение является адаптивным, это

достигается при помощи Bootstrap. При реализации проекта были задействованы шаблоны проектирования MVC и DAO. Также его написание осуществлялось в соответствии с принципами SOLID, DI и IoC. Запуск проекта осуществляется в контейнере Docker, запущенном в режиме swarm (роя). За счёт этого осуществляется отказоустойчивость.

Шаблон проектирования MVC подразумевает распределение кода по модулям, каждый из которых отвечает за решение отведённых ему задач. Данные модули называются `package` (пакеты):

— Controller — содержит все контроллеры проекта
— Model — содержит основные модели проекта, структуры данных используемые в нём для хранения информации
— View — содержит все представления проекта

Для работы сервиса также необходим доступ к базе данных. Он реализован посредством паттерна DAO. Взаимодействие с базой осуществляется через ORM Hibernate.

Главная страница приложения выглядит так:

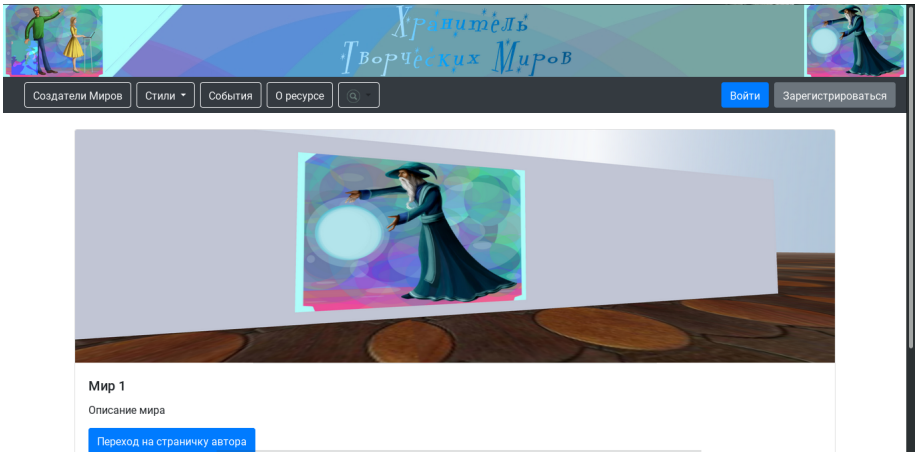

Рисунок 2 – Главная страница приложения-сайта

На ней есть кнопки регистрации и входа в систему. Также на ней изображаются фотографии миров с их кратким описанием.

Вот как выглядит виртуальный мир:

Рисунок 3 – Вид 1

Переход в виртуальную реальность осуществляется посредством очков виртуальной реальности и смартфона.

Оптическая система глаз устроена так, что изображения получаемые с каждого глаза преобразуются в одно. Поэтому человек видит одну картинку, а не две. По такому же принципу работает трёхмерное зрение. Окно браузера смартфона разбивается на две одинаковые части с одним и тем же изображением. Смартфон помещается в очки виртуальной реальности и изображения галереи проходя через линзы равномерно передаются на глаза человека и преобразуются в одну картинку виртуального мира.

Рисунок 4 – Окно браузера для перехода в виртуальную реальность

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы было разработано веб-приложение «Хранитель Творческих Миров», представляющее собой виртуализированную галерею миров. Приложение реализовано с использованием таких шаблонов проектирования, как MVC и DAO, принципов организации программных продуктов SOLID, DI и IoC, а также библиотеки позволяющей моделировать трёхмерные сцены — Three.js.

Разработанный проект может быть использован для помощи художникам в публикации их работ и, одновременно, построения виртуального мира дополняющего эти работы, ведь, порой нельзя всё выразить на одном холсте — фантазия не имеет границ.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 WEB приложение [Электронный ресурс]. — URL: http://java-online.ru/java-web.xhtml (Дата обращения 11.05.2018). Загл. с экр. Яз. рус.

2 Dependency Injection with the Spring Framework [Электронный ресурс]. — URL: http://www.vogella.com/tutorials/SpringDependencyInjection/article.html (Дата обращения 27.05.2018). Загл. с экр. Яз. рус.

3 Первые шаги в VR [Электронный ресурс]. — URL: https://medium.com/high-technologies-center/first-steps-in-vr-dbb5990822f6 (Дата обращения 15.05.2018). Загл. с экр. Яз. рус.

4 Intro to Inversion of Control and Dependency Injection with Spring [Электронный ресурс]. — URL: http://www.baeldung.com/inversion-control-and-dependency-injection-in-spring (Дата обращения 22.05.2018). Загл. с экр. Яз. рус.

5 Guides and such [Электронный ресурс]. — URL: http://hibernate.org/orm/documentation/5.3/ (Дата обращения 21.05.2018). Загл. с экр. Яз. рус.

6 Bootstrap [Электронный ресурс]. — URL: https://getbootstrap.com/ (Дата обращения 17.05.2018). Загл. с экр. Яз. рус.

7 Spring Persistence Tutorial [Электронный ресурс]. — URL: http://www.baeldung.com/persistence-with-spring-series/ (Дата обращения 09.05.2018). Загл. с экр. Яз. рус.

8 Spring Bean Scopes [Электронный ресурс]. — URL: https://howtodoinjava.com/spring5/core/spring-bean-scopes-tutorial/ (Дата обращения 16.05.2018). Загл. с экр. Яз. рус.

9 Model View Controller Pattern [Электронный ресурс]. — URL: http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/mvc.html (Дата обращения 03.05.2018). Загл. с экр. Яз. рус.

10 What is docker. — URL: https://www.docker.com/what-docker (Дата обращения 05.05.2018). Загл. с экр. Яз. англ.