

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКАЯ РАБОТА №8**

по дисциплине «Разработка серверных частей интернет-ресурсов»

**Применение фреймворка**

**Студент группы** ИКБО-16-19 Мурадов Н.Н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практической работы** преподаватель Волков М.Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Москва 2021

**Содержание**

[Ход работы 3](#_Toc90429225)

[Цель работы 3](#_Toc90429226)

[Результаты выполнения: 9](#_Toc90429227)

[Вывод 12](#_Toc90429228)

[Ответы на вопросы к практической работе 13](#_Toc90429229)

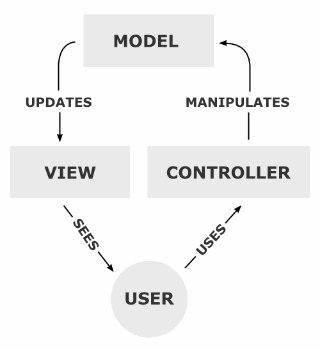
[Список использованной литературы 35](#_Toc90429230)

# Ход работы

# **Цель работы**

В рамках данной практической работы предполагается провести рефакторинг информационной системы, созданной в практических работах 1-7, добавив использование архитектуры MVC (Model-View-Controller). В отчете требуется отразить применение и разработку архитектуры.

В ходе выполнения работы был проведён рефакторинг информационной системы, созданной в практических работах 1-7. В процессе рефакторинга была использована архитектура MVC (см. рисунок 1) (см. Приложение 1).

  
Рисунок 1 - Схема Model – View - Controller.

Для генерации фикстур и рисования графиков были созданы классы, соответствующие слоям MVC: Model, View, Controller. Use Case разделяются соответствующими файлами и моделями.

В качестве фреймворка был выбран SpringFramework [7] в виду того, что он предоставил необходимые пакеты инструментов для реализации данной практической работы.

Краткая сводка:

**Spring Data JDBC (Java Database Connectivity) –** данный пакет инструментов используется для взаимодействия с СУБД. Идея данного инструмента заключается в том, что он в отличие от JPA (Java Persistence API) фокусируется на более простой модели исполнения. Не будет кеширования, отслеживания изменений, или ленивой загрузки. Вместо этого, SQL запросы будут выполнены тогда и только тогда, когда вызывается метод репозитория. Возвращаемый результат будет полностью загружен в память после выполнения метода.

**Spring-session-data-redis** – пакет инструментов, который необходимо было использовать для автоматизированного управления данными о пользовательских сессиях в redis [8].

**Spring-boot-starter-data-redis** – пакет инструментов, упрощённого взаимодействия с NoSQL базой данных redis

**Spring-boot-starter-web** – пакет инструментов, необходимый для создания web-приложения, содержащего в основе структуру REST (Representational State Transfer) [9].

**Spring-boot-starter-security** – пакет инструментов, используемый для построения систем аутентификации и авторизации, а также предоставляет другие возможности безопасности для промышленных приложений.

**Spring-boot-starter-thymeleaf** – набор интеграций, который позволяет его как полнофункциональную замену JSP (JavaServer Pages) [10] в приложениях построенных на архитектуре MVC.

На основе информационной системы, созданной в практических работах 1–7 был произведён переход на поддержку SpringFramework.

Далее можно увидеть примеры листинга кода:

*Листинг* 1 – MvcConfig

*package* com.example.java.configs;  
  
*import* org.springframework.beans.factory.annotation.*Value*;  
*import* org.springframework.context.annotation.*Configuration*;  
*import* org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;  
*import* org.springframework.web.servlet.config.annotation.ViewControllerRegistry;  
*import* org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
  
*@Configuration  
public class* MvcConfig *implements* WebMvcConfigurer {  
 *@Value*("${app.path.upload.file}")  
 *private* String pathUploadFile;  
  
 *@Override  
 public void* addResourceHandlers(ResourceHandlerRegistry registry) {  
 String projectDir = System.getProperty("user.dir");  
 registry  
 .addResourceHandler("/uploads/\*\*")  
 .addResourceLocations("file:/" + projectDir + "/" + pathUploadFile + "/");  
  
 registry  
 .addResourceHandler("/\*\*")  
 .addResourceLocations("classpath:/static/");  
 }  
  
 *public void* addViewControllers(ViewControllerRegistry registry) {  
 registry.addViewController("/login").setViewName("login");  
 }  
}

*Листинг* 2 – AdminController

*package* com.example.java.controller;  
  
*import* com.example.java.dto.UploadPdfDTO;  
*import* com.example.java.modules.InterestingFact;  
*import* com.example.java.modules.User;  
*import* com.example.java.modules.UserInfo;  
*import* com.example.java.service.FileService;  
*import* com.example.java.service.InterestingFactService;  
*import* com.example.java.service.UserService;  
*import* com.example.java.utils.LanguageType;  
*import* com.example.java.utils.ThemeType;  
*import* com.example.java.utils.language.LanguageHelper;  
*import* com.example.java.utils.session.SessionAttribute;  
*import* com.example.java.utils.session.SessionHelper;  
*import* lombok.*RequiredArgsConstructor*;  
*import* org.springframework.beans.factory.annotation.*Value*;  
*import* org.springframework.stereotype.*Controller*;  
*import* org.springframework.ui.Model;  
*import* org.springframework.web.bind.annotation.*DeleteMapping*;  
*import* org.springframework.web.bind.annotation.*GetMapping*;  
*import* org.springframework.web.bind.annotation.*ModelAttribute*;  
*import* org.springframework.web.bind.annotation.*PathVariable*;  
*import* org.springframework.web.bind.annotation.*PostMapping*;  
*import* org.springframework.web.bind.annotation.*RequestMapping*;  
*import* org.springframework.web.multipart.MultipartFile;  
  
*import* javax.servlet.http.Cookie;  
*import* javax.servlet.http.HttpServletResponse;  
*import* javax.servlet.http.HttpSession;  
  
*@Controller  
@RequestMapping*(value = "/admin")  
*@RequiredArgsConstructor  
public class* AdminController {  
 *private final* FileService fileService;  
 *private final* LanguageHelper languageHelper;  
 *private final* UserService userService;  
 *private final* InterestingFactService interestingFactService;  
  
 *@Value*("${graphic.access.key}")  
 *private* String graphicAccessKey;  
  
 *@GetMapping  
 public* String index(HttpSession session, Model model) {  
 SessionHelper sessionHelper = *new* SessionHelper(session);  
  
 LanguageType language = (LanguageType) sessionHelper.getAttribute(SessionAttribute.LANGUAGE);  
 ThemeType theme = (ThemeType) sessionHelper.getAttribute(SessionAttribute.THEME);  
 String name = (String) sessionHelper.getAttribute(SessionAttribute.NAME);  
  
 model.addAttribute("userInfo", *new* UserInfo());  
  
 model.addAttribute("sessionLanguage", language);  
 model.addAttribute("nickname", name);  
 model.addAttribute("sessionTheme", theme);  
  
 model.addAttribute("files", fileService.getFilesInfo());  
 model.addAttribute("dictionary", languageHelper.getDictionary(language));  
  
 model.addAttribute("users", userService.selectAll());  
 model.addAttribute("interestingFacts", interestingFactService.selectAll());  
  
 *return* "admin";  
 }  
  
 *@GetMapping*("/graphics")  
 *public* String graphics(HttpServletResponse response) {  
 Cookie cookie = *new* Cookie("graphic\_access\_key", graphicAccessKey); *// ключ для графиков* cookie.setSecure(*true*);  
 cookie.setHttpOnly(*true*);  
  
 response.addCookie(cookie);  
  
 *return* "graphics";  
 }  
  
 *@PostMapping*("/setting")  
 *public* String setUserInfo(*@ModelAttribute* UserInfo userInfo, HttpSession session) {  
 SessionHelper sessionHelper = *new* SessionHelper(session);  
  
 sessionHelper.setAttribute(SessionAttribute.LANGUAGE, userInfo.getLanguage());  
 sessionHelper.setAttribute(SessionAttribute.NAME, userInfo.getName());  
 sessionHelper.setAttribute(SessionAttribute.THEME, userInfo.getTheme());  
  
 *return* "redirect:/admin";  
 }  
  
 *@PostMapping*("/pdf")  
 *public* String uploadPdf(*@ModelAttribute* UploadPdfDTO uploadPdfDTO) {  
 MultipartFile userFile = uploadPdfDTO.getUserfile();  
  
 *if* (userFile.getContentType().equals("application/pdf")) {  
 fileService.saveFile(userFile);  
 }  
  
 *return* "redirect:/admin";  
 }  
  
 *@PostMapping*("/users")  
 *public* String insertUser(*@ModelAttribute* User user) {  
 userService.insert(user);  
  
 *return* "redirect:/admin";  
 }  
  
 *@DeleteMapping*("/users/{id}")  
 *public* String deleteUser(*@PathVariable* Integer id) {  
 userService.delete(id);  
  
 *return* "redirect:/admin";  
 }  
  
 *@PostMapping*("/interesting\_fact")  
 *public* String insertInterestingFact(*@ModelAttribute* InterestingFact interestingFact) {  
 interestingFactService.insert(interestingFact);  
  
 *return* "redirect:/admin";  
 }  
  
 *@DeleteMapping*("/interesting\_fact/{id}")  
 *public* String deleteInterestingFact(*@PathVariable* Integer id) {  
 interestingFactService.delete(id);  
  
 *return* "redirect:/admin";  
 }  
}

*Листинг* 3 – FileRepository

*package* com.example.java.repository;  
  
*import* com.example.java.modules.FileInfo;  
*import* com.example.java.modules.InterestingFact;  
*import* com.example.java.modules.UserInfo;  
*import* lombok.*RequiredArgsConstructor*;  
*import* org.springframework.jdbc.core.BeanPropertyRowMapper;  
*import* org.springframework.jdbc.core.JdbcTemplate;  
*import* org.springframework.jdbc.support.GeneratedKeyHolder;  
*import* org.springframework.jdbc.support.KeyHolder;  
*import* org.springframework.stereotype.*Repository*;  
  
*import* java.math.BigInteger;  
*import* java.sql.PreparedStatement;  
*import* java.sql.Statement;  
*import* java.util.List;  
*import* java.util.Map;  
  
*@Repository  
@RequiredArgsConstructor  
public class* FileRepository {  
 *private final* JdbcTemplate jdbcTemplate;  
  
 *public* List<FileInfo> selectAll() {  
 String SQL = "SELECT \* FROM files\_info";  
 *return* jdbcTemplate.query(SQL, *new* BeanPropertyRowMapper<>(FileInfo.*class*));  
 }  
  
 *public* Integer insert(FileInfo fileInfo) {  
 String SQL = "INSERT INTO files\_info (name) VALUES (?)";  
 KeyHolder key = *new* GeneratedKeyHolder();  
  
 jdbcTemplate.update(  
 connection -> {  
 PreparedStatement ps = connection.prepareStatement(SQL, Statement.RETURN\_GENERATED\_KEYS);  
 ps.setString(1, fileInfo.getName());  
 *return* ps;  
 },  
 key);  
  
 *return* ((BigInteger) key.getKeyList().get(0).get("GENERATED\_KEY")).intValue();  
 }  
}

*Листинг* 4 – FileService

*package* com.example.java.service;  
  
*import* com.example.java.modules.FileInfo;  
*import* com.example.java.repository.FileRepository;  
*import* lombok.*RequiredArgsConstructor*;  
*import* org.springframework.beans.factory.annotation.*Value*;  
*import* org.springframework.stereotype.*Service*;  
*import* org.springframework.web.multipart.MultipartFile;  
  
*import* javax.annotation.*PostConstruct*;  
*import* java.io.File;  
*import* java.io.IOException;  
*import* java.nio.file.Files;  
*import* java.nio.file.Path;  
*import* java.nio.file.Paths;  
*import* java.util.ArrayList;  
*import* java.util.List;  
*import* java.util.Objects;  
  
*@Service  
@RequiredArgsConstructor  
public class* FileService {  
 *private final* FileRepository fileRepository;  
  
 *@Value*("${app.path.upload.file}")  
 *private* String uploadPath;  
  
 *@PostConstruct  
 public void* init() {  
 *try* {  
 Files.createDirectories(Paths.get(uploadPath));  
 } *catch* (IOException e) {  
 *throw new* RuntimeException("Could not create upload folder!");  
 }  
 }  
  
 *public void* saveFile(MultipartFile file) {  
 *try* {  
 Path root = Paths.get(uploadPath);  
 *if* (!Files.exists(root)) {  
 init();  
 }  
  
 FileInfo fileInfo = *new* FileInfo();  
 fileInfo.setName(file.getOriginalFilename());  
  
 Integer id = fileRepository.insert(fileInfo);  
  
 Files.copy(file.getInputStream(), root.resolve(id.toString() + ".pdf"));  
 } *catch* (Exception e) {  
 *throw new* RuntimeException("Could not store the file. Error: " + e.getMessage());  
 }  
 }  
  
 *public* List<FileInfo> getFilesInfo() {  
 *return* fileRepository.selectAll();  
 }  
}

*Листинг* 5 – docker-compose.develop.yml

version: '3'  
  
services:  
 proxy:  
 image: nginx:1.11  
 ports:  
 - '80:80'  
 volumes:  
 - ./nginx.develop.conf:/etc/nginx/conf.d/default.conf:ro  
 php:  
 build:   
 context: .  
 dockerfile: php.Dockerfile  
 volumes:  
 - ./php:/var/www/html/php  
 restart: always  
 db:  
 image: 'mysql:latest'  
 volumes:   
 - ./init.sql:/data/application/init.sql  
 command: --init-file /data/application/init.sql  
 restart: always  
 ports:  
 - 3306:3306  
 environment:   
 MYSQL\_ROOT\_PASSWORD: root  
 adminer:  
 image: adminer  
 restart: always  
 ports:  
 - 8081:8080  
 redis:  
 image: "redis:alpine"  
 ports:  
 - 6379:6379

*Листинг* 6 – docker-compose.production.yml

version: '3'  
  
services:  
 proxy:  
 image: nginx:1.11  
 ports:  
 - '80:80'  
 volumes:  
 - ./nginx.production.conf:/etc/nginx/conf.d/default.conf:ro  
 php:  
 build:   
 context: .  
 dockerfile: php.Dockerfile  
 volumes:  
 - ./php:/var/www/html/php  
 restart: always  
 java:  
 build:   
 context: .  
 dockerfile: java.Dockerfile  
 restart: always  
 db:  
 image: 'mysql:latest'  
 volumes:   
 - ./init.sql:/data/application/init.sql  
 command: --init-file /data/application/init.sql  
 restart: always  
 environment:   
 MYSQL\_ROOT\_PASSWORD: root  
 adminer:  
 image: adminer  
 restart: always  
 ports:  
 - 8081:8080  
 redis:  
 image: "redis:alpine"

# Результаты выполнения:

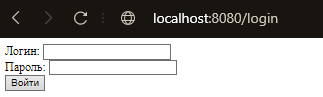


Рисунок 2 – Авторизация

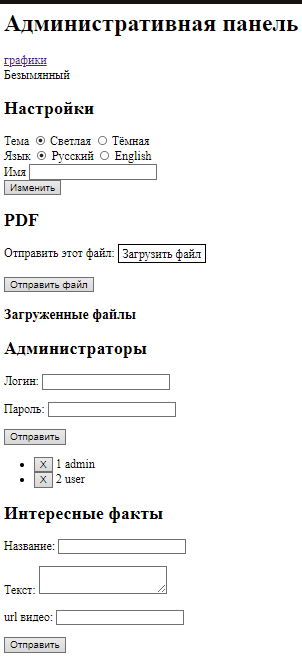


Рисунок 3 – Вход в административную панель

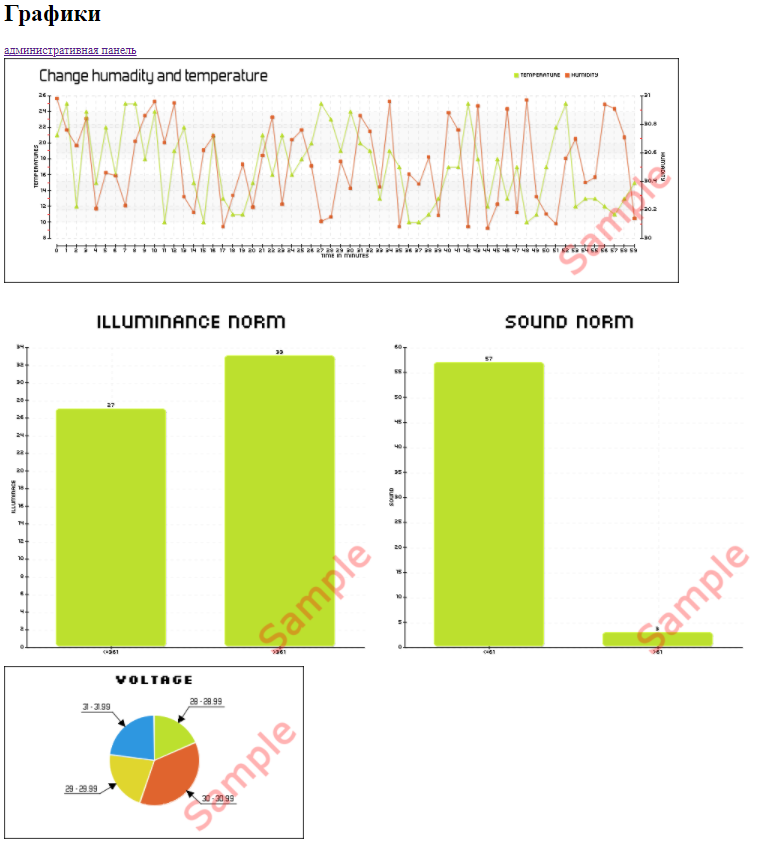


Рисунок 4 – Результат создания фикстур и отображение графиков

# Вывод

В результате выполнения данной практической работы был произведён рефакторинг информационной системы, созданной в практических работах 1-7. Была внедрена архитектура Model-View-Controller с использованием SpringFramework.

# Ответы на вопросы к практической работе

1. Что такое фреймворк?

Программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

1. Что такое фреймворк и чем он отличается от библиотеки?

«Фреймворк» отличается от понятия библиотеки тем, что библиотека может быть использована в программном продукте просто как набор подпрограмм близкой функциональности, не влияя на архитектуру программного продукта и не накладывая на неё никаких ограничений. В то время как «фреймворк» диктует правила построения архитектуры приложения, задавая на начальном этапе разработки поведение по умолчанию — «каркас», который нужно будет расширять и изменять, согласно указанным требованиям.

Также, в отличие от библиотеки, которая объединяет в себе набор близкой функциональности, — «фреймворк» может содержать в себе большое число разных по тематике библиотек.

Другим ключевым отличием «фреймворка» от библиотеки может быть инверсия управления: пользовательский код вызывает функции библиотеки (или классы) и получает управление после вызова. Во «фреймворке» пользовательский код может реализовывать конкретное поведение, встраиваемое в более общий — «абстрактный» код фреймворка. При этом «фреймворк» вызывает функции (классы) пользовательского кода

1. Из каких этапов состоит разработка веб-приложения?

Планирование, проектирование, тестирование, запуск.

1. Что такое “Гибкий” фреймворк?

Фреймворк, следующий гибкой методологии разработки.

1. Что такое “не гибкий” фреймворк?

Фреймворк, не следующий гибкой методологии разработки.

1. Чем “гибкий” фреймворк отличается от “не гибкого” фреймворка?

В основном – разные жизненные циклы приложения. Большая возможность соответствия декларируемым в agile-манифесте принципам.

1. Опишите фреймворк Laravel.

Бесплатный веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC.

1. Какие приложения можно создавать с помощью Laravel?

С помощью Laravel, как и на базе любого другого фреймворка, можно делать абсолютно разные типы сайтов, начиная с лендингов и заканчивая социальными сетями.

1. Что такое Eloquent ORM?

Система объектно-реляционного отображения (ORM) Eloquent — красивая и простая реализация шаблона ActiveRecord в Laravel для работы с базами данных. Каждая таблица имеет соответствующий класс-модель, который используется для работы с этой таблицей. Модели позволяют запрашивать данные из таблиц, а также вставлять в них новые записи.

1. Что такое сервис-провайдеры (service providers)?

Сервис-провайдеры лежат в основе первоначальной загрузки всех приложений на Laravel. И ваше приложение, и все базовые сервисы Laravel загружаются через сервис-провайдеры.

Но что мы понимаем под "первоначальной загрузкой"? В общих чертах, мы имеем ввиду регистрацию таких вещей, как биндингов в IoC-контейнер (фасадов и т.д.), слушателей событий, фильтров роутов и даже самих роутов. Сервис-провайдеры - центральное место для конфигурирования вашего приложения.

Если вы откроете файл config/app.php, поставляемый с Laravel, то увидите массив providers. В нём перечислены все классы сервис-провайдеров, которые загружаются для вашего приложения. Конечно, многие из них являются "отложенными" провайдерами, т.е. они не загружаются при каждом запросе, а только при необходимости.

1. Что такое Сервис-контейнер в Laravel?

Сервис-контейнер в Laravel — это мощное средство для управления зависимостями классов и внедрения зависимостей. Внедрение зависимостей — это распространенный термин, который означает добавление других классов в этот класс через конструктор или, в некоторых случаях, метод-сеттер.

1. Что такое Контракты в Laravel?

Контракты в Laravel — это набор интерфейсов, которые описывают основной функционал, предоставляемый фреймворком. Например, контракт Illuminate\Contracts\Queue\Queue определяет методы, необходимые для организации очередей, в то время как контракт Illuminate\Contracts\Mail\Mailer определяет методы, необходимые для отправки электронной почты.

1. Что такое Фасады в Laravel?

Фасады предоставляют "статический" интерфейс к классам, доступным в сервис-контейнере. Laravel поставляется со множеством фасадов, которые предоставляют доступ практически ко всем функциям Laravel. Фасады Laravel служат "статическими прокси" для основополагающих классов в сервис-контейнере, предоставляя преимущество лаконичного, выразительного синтаксиса, сохраняя при этом большую тестируемость и гибкость по сравнению с обычными статическими методами.

1. Опишите структуру Laravel-приложения по умолчанию.

* Корневой каталог
  + Каталог app
  + Каталог bootstrap
  + Каталог config
  + Каталог database
  + Каталог public
  + Каталог resources
  + Каталог routes
  + Каталог storage
  + Каталог tests
  + Каталог vendor
* Каталог пространства App
  + Каталог пространства Broadcasting
  + Каталог пространства Console
  + Каталог пространства Events
  + Каталог пространства Exceptions
  + Каталог пространства Http
  + Каталог пространства Jobs
  + Каталог пространства Listeners
  + Каталог пространства Mail
  + Каталог пространства Models
  + Каталог пространства Notifications
  + Каталог пространства Policies
  + Каталог пространства Providers
  + Каталог пространства Rules

1. Опишите преимущества фреймворка Laravel.

Достаточно неплохая и понятная документация.

Вокруг фреймворка создана мощная экосистема. Различные курсы, конференции, обучающие материалы позволяют собрать вокруг фреймворка большое количество разработчиков и спонсоров, которые заинтересованы в развитии инструмента и принимают в этом участие. Да, здесь чувствуется запах маркетинга, и неплохой.

Одним из самых очевидных плюсов Laravel, является гибкая система маршрутизации, позволяющая составить самые разные проверки маршрута веб-приложения. Вы можете выделить маршруты в специальные группы, использовать пространство имен, указать параметры маршрута, использовать регулярные выражения, настроить поддоменную маршрутизацию и многое другое.

В Laravel много синтаксического сахара. Синтаксис API фреймворка достаточно простой и понятный. Здесь нет длинных и сложных конструкций, а только краткие и продуманные названия функций.

Laravel содержит удобный механизм обработки ошибок и исключений.

Фреймворк включает в себя встроенные механизмы аутентификации и авторизации пользователей, которую можно перенастроить под свои потребности.

Laravel предоставляет из коробки механизмы для кэширования веб-приложения с помощью Memcached и Redis. Кроме этого есть удобные функции для использования простого файлового кэширования данных.

Laravel предоставляет чистый и простой API поверх популярной библиотеки SwiftMailer с драйверами для SMTP, Mailgun, SparkPost, Amazon SES и sendmail, чтобы сделать отправку почты через локальную или облачную службу по выбору. В том числе есть механизм для построения очередей отправки почты.

Laravel Cashier обеспечивает выразительный, свободный интерфейс к сервисам биллинга по подписке Stripe и Braintree.

1. Опишите недостатки фреймворка Laravel.

Синтаксический сахар в Laravel как плюс, так может быть и минусом. Очень легко привыкнуть к нему и позабыть, как пишутся чистые запросы и функции.

Нарушение обратная совместимости между версиями фреймворка.

Не логичное расположение каталогов и файлов. Например, по умолчанию в прямо в каталоге /app расположена модель User.php, которую логичней было бы расположить в каталоге /app/Models. Каталог resources с файлами представления размещен в корне приложения, хотя логичней будет его разместить в /app/resources.

1. Что включает в себя фреймворк Laravel.

 Laravel включает в себя встроенную поддержку для аутентификации, локализации, модели, представления, сессий, маршрутизации и других механизмов, поддержку контроллеров, которые сделали фреймворк полностью MVC-совместимым, встроенную поддержку для инверсии управления и шаблонизатор Blade, интерфейс командной строки (CLI) под именем «Artisan», встроенную поддержку нескольких систем управления базами данных, миграции баз данных в виде контроля версий, обработку событий, выгрузка таблиц базы данных для первоначальной популяции, поддержку очередей сообщений, встроенную поддержку отправки различных типов электронной почты и поддержку «мягкого» удаления записей базы данных, поддержку планирования периодически выполняемых задач через пакет Scheduler, слой абстракции Flysystem, который позволяет использовать удаленное хранилище так же, как и локальные файловые системы, улучшенную обработку активов пакета через Elixir и упрощенная аутентификация с внешней стороны через дополнительный пакет Socialite, Laravel Dusk, Laravel Mix, Blade Components и Slots, Markdown Emails, автоматические фасады, улучшения маршрута.

1. Опишите из чего состоит экосистема Laravel.

PHP, Composer, Dotenv, PSR-4, Eloquent ORM, Flysystem, Elixir, HHVM, Homestead, Rocketeer

1. Опишите фреймворк Symfony.

Свободный PHP фреймворк для быстрой разработки веб-приложений и решения рутинных задач веб-программистов. Разработка и поддержка фреймворка спонсируется французской компанией Sensio.

Symfony состоит из набора не связанных между собой компонентов, которые можно использовать повторно в проектах.

Symfony позволяет устанавливать сторонние пакеты, библиотеки, компоненты и настраивать их с помощью конфигурации в форматах YAML, XML, PHP, а также .env файлах.

Symfony не обеспечивает компонент для работы с базой данных, но обеспечивает тесную интеграцию с библиотекой Doctrine.

Symfony предоставляет функцию почтовой программы на основе популярной библиотеки Swift Mailer. Эта почтовая программа поддерживает отправку сообщений с ваших собственных почтовых серверов, а также с использованием популярных почтовых провайдеров, таких как Mandrill, SendGrid и Amazon SES.

Механизм интернационализации позволяет установить и произвести перевод сообщений веб-приложения на основе выбранного языка или страны.

Symfony предлагает систему логирования ошибок приложения, а также подключить библиотеку логирования Monolog.

1. Опишите преимущества фреймворка Symfony.

Мощная экосистема вокруг фреймворка, с хорошим сообществом и множеством разработчиков.

Хорошая и постоянно обновляемая документация для всех версий фреймворка.

Множество различных не связанных компонентов для повторного использования.

Предлагает механизм функциональных и модульных тестов для нахождения ошибок в веб-приложении.

Подходит для сложных и нагруженных веб-проектов.электронной коммерции.

1. Опишите недостатки фреймворка Symfony.

Несмотря на хорошую документацию, фреймворк является сложным для изучения.

1. Опишите фреймворк Code Igniter.

Это популярный PHP микро-фреймворк с открытым исходным кодом, для разработки веб-систем и приложений.

В CodeIgniter компоненты загружаются и процедуры выполняются только по запросу, а не глобально. Система не делает никаких предположений относительно того, что может потребоваться помимо минимальных основных ресурсов, поэтому система по умолчанию очень легкая.

Компоненты фреймворка слабо связаны между собой и не зависят друг от друга. Чем меньше компонентов зависит друг от друга, тем более гибкой и многоразовой становится система.

Хотя CodeIgniter работает довольно быстро, объем динамической информации, отображаемой на страницах, будет напрямую зависеть от используемых ресурсов сервера, памяти и циклов обработки, которые влияют на скорость загрузки страниц.

Поэтому CodeIgniter позволяет кэшировать страницы для достижения максимальной производительности. с помощью встроенного компонента кэширования.

1. Опишите преимущества фреймворка Code Igniter.

Отличная документация и англоязычное сообщество.

Высокая производительность фреймворка.

Небольшой размер фреймворка.

Предоставляет легкие и простые решения для разработки.

Подходит для быстрой разработки небольших сайтов и веб-приложений.

Структура фреймворка не требует строгих правил кодирования.

Не требует сложной настройки, почти нулевая конфигурация.

MVC-архитектура веб-приложения.

Слабая связанность компонентов.

Множество подключаемых библиотек и помощников.

1. Опишите недостатки фреймворка Code Igniter.

Задержка в развитии и переходе на новые технологии.

1. Опишите фреймворк Yii2.

Объектно-ориентированный компонентный фреймворк для PHP, реализующий парадигму MVC (Model-View-Controller). Yii является акронимом от “Yes It is”, на русском пишется и читается как “йии”. Yii2 является второй версией фреймворка Yii.

26. Опишите преимущества фреймфорка Yii2.

Фреймворк прост в понимании.

Легко адаптируется под большие и маленькие проекты.

Имеет большое количество решений рутинных задач из коробки. К примеру, шаблон advanced обладает механизмом авторизации и аутентификации. Это довольно нужный механизм и он не очень прост в реализации.

Имеет замечательную документацию, гайды по старту и различные рецепты.

Yii2 популярен и довольно стар (релиз-то был аж в 2014 году), поэтому на рынке в много вакансий yii2-разработчиков, а с помощью развитого сообщества ответы на 90% вопросов вы найдете при легком гуглеже.

С помощью шаблонов и gii фреймворк подсказывает начинающему разработчику, как правильно располагать файлы. Сначала разработчик начинает повторять за тем, как это сделано в фреймворке, а потом понимает почему это хорошо. Разработчики фреймворка будто делятся опытом с новичком

27. Опишите Недостатки фреймворка Yii2.

Наличие различных антипаттернов в проекте — например, одиночка или божественный объект.

Встроенный класс User, являющийся потомком от ActiveRecord, показывает, как делать не нужно. Учит начинающих программистов, что классы, наследуемые от ActiveRecord, нужно раздувать различными методами, не связанными с работой с БД.

Сильная связность модулей в приложении. Говорят, эта проблема в Yii3 будет решена

Медленное развитие.

1. Опишите экосистему Spring.

Spring Boot, Spring MVC, Spring Web Flow, Spring Web Services, Spring Security, Spring Integration, Spring Batch, Spring Social, Spring Mobile, Spring Dynamic Modules, Spring LDAP, Spring Rich Client, Spring.NET, Spring-Flex, Spring Roo

1. Что такое Spring MVC?

Это веб-фреймворк Spring. Он позволяет создавать веб-сайты или RESTful сервисы (например, JSON/XML) и хорошо интегрируется в экосистему Spring, например, он поддерживает контроллеры и REST контроллеры в ваших Spring Boot приложениях.

1. Что такое Spring Boot?

Spring Boot — это полезный проект, целью которого является упрощение создания приложений на основе Spring. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода.

1. Определите особенности Spring Boot и его роль в разработке приложений с помощью фреймворка Spring?

Если Spring Framework фокусируется на предоставлении гибкости, то Spring Boot стремится сократить длину кода и упростить разработку web-приложения. Используя конфигурацию при помощи аннотаций и стандартного кода, Spring Boot сокращает время, затрачиваемое на разработку приложений.

1. Опишите процесс управления зависимостями с помощью Spring Boot.

Spring Boot решает эту проблему путём предоставления набора зависимостей, облегчая жизнь разработчикам. Например, если вы желаете использовать Spring и JPA в целях доступа к базе данных, вам достаточно просто включить в проект зависимость spring-boot-starter-data-jpa.

1. Опишите процесс автоматической конфигурации на Spring Boot.

Spring Boot автоматическая настройка пытается автоматически настроить приложение Spring на основе добавленных зависимостей jar. Например, если HSQLDB находится на вашем classpath, и вы не настроили вручную никаких компонентов подключения к базе данных, то автоматически настроится база данных в памяти.

1. Что такое HttpServlet?

Сервлет является интерфейсом Java, реализация которого расширяет функциональные возможности сервера. Сервлет взаимодействует с клиентами посредством принципа запрос-ответ.

Хотя сервлеты могут обслуживать любые запросы, они обычно используются для расширения веб-серверов. Для таких приложений технология Java Servlet определяет HTTP-специфичные сервлет классы.

1. Что делает Spring MVC DispatcherServlet?

Это главный контроллер в приложении Spring MVC, который обрабатывает все входящие запросы и передает их для обработки в различные методы в контроллеры.

1. Что такое ViewResolver?

Интерфейс, реализуемый объектами, которые способны находить представления View по имени View Name.

1. Опишите REST контроллеры.

Когда вы разрабатываете RESTFul сервисы, все немного по-другому. Ваш клиент, будь то браузер или другой веб-сервис, будет (обычно) создавать запросы JSON или XML. Клиент отправляет, скажем, запрос JSON, вы обрабатываете его, а затем отправитель ожидает возврата JSON.

Но на стороне Java (в вашей программе Spring MVC) вы не хотите иметь дело с JSON строками. Ни при получении запросов, как указано выше, ни при отправке ответов обратно клиенту. Вместо этого вы хотели бы просто иметь объекты Java, в которые Spring автоматически конвертирует JSON. i.e. DTO

Это также означает, что вам *не нужна* вся эта обработка модели и представления, которые вам приходилось делать при рендеринге HTML в ваших контроллерах. Для RESTful сервисов у вас нет библиотеки шаблонов, читающей шаблон HTML и заполняющей его данными модели, чтобы сгенерировать для вас ответ JSON.

Вместо этого вы хотите перейти непосредственно из HTTP запрос → Java объект и из Java объект → HTTP ответ.

Как вы уже догадались, это именно то, что Spring MVC обеспечивает при написании REST контроллера.

1. Что такое HttpMessageConverter?

HttpMessageConverter — это интерфейс с четырьмя методами (обратите внимание, я немного упростил интерфейс для более простого объяснения, так как он выглядит немного более продвинутым в реальной жизни).

* canRead (MediaType) → Может ли этот конвертер читать (JSON | XML | YAML | и т. д.)? Переданный здесь MediaType обычно является значением из заголовка запроса Content-Type.
* canWrite (MediaType) → Может ли этот преобразователь писать (JSON | XML | YAML | и т. д.)? Тип MediaType, переданный здесь, обычно является значением из заголовка запроса Accept.
* read(Object, InputStream, MediaType) → Читать мой Java-объект из (JSON | XML | YAML | и т. д.) InputStream
* write(Object, OutputStream, MediaType) → Записать мой Java-объект в OutputStream как (JSON | XML | YAML | и т. д.)

Короче говоря, MessageConverter должен знать, какие MediaTypes он поддерживает (например, application/json), а затем должен реализовать два метода для фактического чтения / записи в этом формате данных.

1. Какие есть HttpMessageConverters?

AllEncompassingFormHttpMessageConverter

1. В чем разница между Spring MVC и Spring Boot?

Spring MVC – это полный HTTP-ориентированный MVC-фреймворк, управляемый Spring Framework и основанный на сервлетах. Spring boot - это утилита для быстрой настройки приложений, предлагающая готовую конфигурацию для создания приложений на базе Spring. / «Нет никакой разницы, Spring Boot использует и строит приложение поверх Spring MVC.»

1. Какой тип ввода HTTP-запроса понимает Spring MVC?

Spring MVC понимает практически все, что предлагает HTTP — с помощью сторонних библиотек.

Это означает, что вы можете добавить в него тела запросов JSON, XML или HTTP (Multipart) Fileuploads, и Spring будет удобно конвертировать этот ввод в объекты Java.

1. Какие HTTP-ответы может создавать Spring MVC?

Spring MVC может записывать все что угодно в HttpServletResponse — с помощью сторонних библиотек.

Будь то HTML, JSON, XML или даже тела ответов WebSocket. Более того, он берет ваши объекты Java и генерирует эти тела ответов для вас.

1. В чем разница между контроллером и REST контроллером?

Контроллер по умолчанию возвращают HTML пользователям с помощью библиотеки шаблонов, если вы не добавите аннотацию @ResponseBody к определенным методам, которые также позволяют возвращать XML / JSON.

Исходный код REST контроллера показывает, что на самом деле это контроллер с добавленной аннотацией @ResponseBody. Что эквивалентно написанию контроллера с аннотацией @ResponseBody для каждого метода.

1. Как получить доступ к текущей HttpSession пользователя?

В Spring MVC контроллере или REST контроллере вы можете просто указать HttpSession в качестве аргумента метода, и Spring автоматически вставит его (создав его, если он еще не существует).

1. Как получить доступ к HttpServletRequest?

В вашем Spring MVC контроллере или REST контроллере вы можете просто указать HttpServletRequest в качестве аргумента метода, и Spring автоматически вставит его (создавая, если он еще не существует)

1. Как читать HTTP заголовки?

Существует множество способов получить доступ к заголовкам запросов, в зависимости от того, хотите ли вы только один или карту со всеми из них. В любом случае вам нужно аннотировать их с помощью @RequestHeader.

1. Как получить IP-адрес пользователя?

Это вопрос с подвохом. Существует метод с именем httpServletRequest.getRemoteAddr(), который, однако, возвращает только IP-адрес пользователя или последнего прокси-сервера, отправившего запрос, в 99,99% случаев это ваш Nginx или Apache.

Следовательно, вам нужно проанализировать заголовок X-Forwarded-For для получения правильного IP-адреса. Но что произойдет, если ваше приложение, кроме того, будет работать за CDN, например CloudFront? Тогда ваш X-Forwarded-For будет выглядеть так:

X-Forwarded-For: MaybeSomeSpoofedIp, realIp, cloudFrontIp

Проблема в том, что вы не можете прочитать заголовок слева направо, поскольку пользователи могут предоставить и, следовательно, подделать свой собственный заголовок X-Forwarded-For. Вам всегда нужно идти справа налево и исключать все известные IP-адреса. В случае CloudFront это означает, что вам необходимо знать диапазоны IP-адресов CloudFront и удалить их из заголовка. Ага!

Это приводит к довольно сложному коду, разрешающему IP. Угадайте, сколько проектов сделали это неправильно!

package com.marcobehler.springmvcarticle;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

import javax.servlet.http.HttpServletRequest;

@RestController

public class IpController {

private static final String[] HEADERS\_TO\_TRY = {

"X-Forwarded-For",

"Proxy-Client-IP",

"WL-Proxy-Client-IP",

"HTTP\_X\_FORWARDED\_FOR",

"HTTP\_X\_FORWARDED",

"HTTP\_X\_CLUSTER\_CLIENT\_IP",

"HTTP\_CLIENT\_IP",

"HTTP\_FORWARDED\_FOR",

"HTTP\_FORWARDED",

"HTTP\_VIA",

"REMOTE\_ADDR"};

@GetMapping("/ip")

public String getClientIpAddress(HttpServletRequest request) {

for (String header : HEADERS\_TO\_TRY) {

String ip = request.getHeader(header);

if (ip != null && ip.length() != 0 && !"unknown".equalsIgnoreCase(ip)) {

return getRealClientIpAddress(ip);

}

}

return request.getRemoteAddr();

}

/\*\*

\* Goes through the supplied ip string (could be one or multiple). Traverses it through the right side...

\* and removes any known ip address ranges

\*

\* @param ipString

\* @return

\*/

public String getRealClientIpAddress(String ipString) {

String[] manyPossibleIps = ipString.split(",");

for (int i = manyPossibleIps.length - 1; i >= 0; i--) {

String rightMostIp = manyPossibleIps[i].trim();

if (isKnownAddress(rightMostIp)) {

continue; // skip this ip as it is trusted

} else {

return rightMostIp;

}

}

return ipString;

}

private boolean isKnownAddress(String rightMostIp) {

// do your check here..for cloudfront you'd need to download their ip address ranges

// from e.g. http://d7uri8nf7uskq.cloudfront.net/tools/list-cloudfront-ips

// and compare the current ip against them

return false;

}

}

1. Опишите фреймворк Django.

Свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий фреймворка от некоторых других (например, Ruby on Rails). Один из основных принципов фреймворка — DRY (англ. Don’t repeat yourself).

1. Какое программное обеспечение позволяет разрабатывать django?

Django Rest Framework (DRF) — это библиотека, которая работает со стандартными моделями Django для создания гибкого и мощного API для проекта.

1. Из каких частей состоит обычный проект на Django.

Веб-приложение или проект Django состоит из отдельных приложений. Вместе они образуют полноценное веб-приложение. Каждое приложение представляет какую-то определенную функциональность или группу функциональностей. Один проект может включать множество приложений. Это позволяет выделить группу задач в отдельный модуль и разрабатывать их относительно независимо от других. Кроме того, мы можем переносить приложение из одного проекта в другой независимо от другой функциональности проекта.

При создании проекта он уже содержит несколько приложений по умолчанию.

* django.contrib.admin
* django.contrib.auth
* django.contrib.contenttypes
* django.contrib.sessions
* django.contrib.messages
* django.contrib.staticfiles

1. Что такое Отображения (views) в рамках Django.

*Представления (views)* — центральные персонажи Web-приложений на основе Django.

В Django используются два вида представлений:

1. *Представления-функции (view functions)*,
2. *Представления-классы (class based views)*.

Рассмотрим сначала более простой (но не менее мощный) вид - обычные функции, принимающие на входе *запрос* (объект класса HttpRequest) и возвращающие *ответ* (объект класса HttpResponse или ему подобных).

Все представления этого вида наследуются от класса django.views.View. Метод get здесь работает точь-в-точь как view function. При этом на каждый запрос будет создан новый экземпляр этого класса, так что вы смело можете объявлять в классе методы, которые по ходу выполнения запроса будут менять его состояние. Регистрируется представление-класс с помощью метода (метода класса) as\_view.

1. Что такое модель в рамках Django.

Веб-приложения Django получают доступ и управляют данными через объекты Python, называемые моделями. Модели определяют структуру хранимых данных, включая типы полей и, возможно, их максимальный размер, значения по умолчанию, параметры списка выбора, текст справки для документации, текст меток для форм и т.д.

1. Как производится вывод данных с помощью Django.

Да.

1. Как работает Административная панель в Django.

Она использует мета-данные модели чтобы предоставить многофункциональный, готовый к использованию интерфейс для работы с содержимым сайта.

1. Опишите реализацию аутентификации на Django.

Внутри locallibrary/locallibrary/settings.py:

INSTALLED\_APPS = [

...

'django.contrib.auth', # Фреймворк аутентификации и моделей по умолчанию.

'django.contrib.contenttypes', # Django контент-типовая система (даёт разрешения, связанные с моделями).

....

MIDDLEWARE = [

...

'django.contrib.sessions.middleware.SessionMiddleware', # Управление сессиями между запросами

...

'django.contrib.auth.middleware.AuthenticationMiddleware', # Связывает пользователей, использующих сессии, запросами.

....

1. Опишите преимущества фреймворка Django.

Когда у вас возникает определенная мысль, трансформировать ее на языке программирования и предать ей реальную форму при помощи Django займет всего несколько минут. То, что Django находится в свободном доступе, дает возможность заметно упростить процесс веб разработки, так как разработчик может сфокусироваться на процессе дизайна и разработке функционала приложения. Таким образом, Django – это идеальный инструмент для стартапов, когда веб дизайн должен отображать концепцию и цели компании.

Быстрота: Django был разработан, чтобы помочь разработчикам создать приложение настолько быстро, на сколько это возможно. Это включает в себя формирование идеи, разработку и выпуск проекта, где Django экономит время и ресурсы на каждом из этих этапов. Таким образом, его можно назвать идеальным решением для разработчиков, для которых вопрос дедлайна стоит в приоритете.

Полная комплектация: Django работает с десятками дополнительных функций, которые заметно помогают с аутентификацией пользователя, картами сайта, администрированием содержимого, RSS и многим другим. Данные аспекты помогают осуществить каждый этап веб разработки.

Безопасность: Работая в Django, вы получаете защиту от ошибок, связанных с безопасностью и ставящих под угрозу проект. Я имею ввиду такие распространенные ошибки, как инъекции SQL, кросс-сайт подлоги, clickjacking и кросс-сайтовый скриптинг. Для эффективного использования логинов и паролей, система пользовательской аутентификации является ключом.

Масштабируемость: фреймворк Django наилучшим образом подходит для работы с самыми высокими трафиками. Следовательно, логично, что великое множество загруженных сайтов используют Django для удовлетворения требований, связанных с трафиком.

Разносторонность: менеджмент контента, научные вычислительные платформы, даже крупные организации – со всем этим можно эффективно справляться при помощи Django.

1. Опишите недостатки фреймворка Django.

Использование шаблона маршрутизации с указанием URL

Django слишком монолитный

Все базируется на ORM Django

Компоненты развертываются совместно

Необходимо умение владеть всей системой для работы.

1. Опишите процесс масштабирования приложения с помощью какого-либо современного фреймворка.

Фреймворки имеют разную сложность и рассчитаны на компании или подразделения разного размера. При этом большинство из них рассчитано на короткие цепочки создания ценности, когда одна кроссфункциональная команда делает продукт, поставляемый потребителям. Короткие цепочки являются естественным способом организации труда, способным быстро реагировать на изменения, в отличие от стабильных условий функционирования, которые ведут к специализации и образованию длинных цепочек из функциональных подразделений.

1. Опишите ситуации, когда для масштабирования системы, написанной с помощью фреймворка требуется реинжиниринг системы.

К примеру, система плохо сынженирована.

1. Опишите как происходит процесс перехода с одного фреймворка на другой.
2. Когда требуется переход с одного фреймворка на другой?

Когда ограничения изначального фреймворка начинают сильно мешать, а возможности изначального фреймворка больше не являются достаточными.

1. Что такое Пространство задач и пространство решений?

Предметные области состоят из пространства задач и пространства решений. Пространство задач позволяет думать о стратегической бизнес проблеме, которая должна быть решена, а пространство решений, сосредоточится на том, как реализуется программное обеспечение, чтобы решить бизнес проблему.

Пространство задач – части предметной области, которые необходимы, чтобы создать смысловое ядро. Это комбинация смыслового ядра и подобластей, которое это ядро должно использовать.

Пространство решений – один или несколько ограниченных контекстов, набор конкретных моделей программного обеспечения. Разработанный ограниченный контекст – это конкретное решение, представление реализации.

Идеальным вариантом является обеспечение однозначного соответствия между подобластями и ограниченными контекстами. Таким образом, объединяются пространство задач и пространство решений, выделяются модели предметной области в четко определенные области в зависимости от поставленных целей. Если система не разрабатывается с нуля, она часто представляет собой большой комок грязи, где подобласти пересекаются с ограниченными контекстами.

# Список использованной литературы

1. Configure LAMP Stack in Docker Containers: [Электронный ресурс]. URL: https://www.centlinux.com/2020/03/configure-lamp-stack-in-docker-containers.html (дата обращения: 02.09.2021)
2. Сейерс, Э. Х. Docker на практике / Э. Х. Сейерс, А. Милл ; перевод с английского Д. А. Беликов. — Москва : ДМК Пресс, 2020. — 516 с. — ISBN 978-5-97060-772-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131719 (дата обращения: 02.09.2021). — Режим доступа: для авториз. пользователей.
3. Моуэт, Э. Использование Docker / Э. Моуэт ; научный редактор А. А. Маркелов ; перевод с английского А. В. Снастина. — Москва : ДМК Пресс, 2017. — 354 с. — ISBN 978-5-97060-426-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/93576 (дата обращения: 02.09.2021). — Режим доступа: для авториз. пользователей.
4. Кочер, П. С. Микросервисы и контейнеры Docker : руководство / П. С. Кочер ; перевод с английского А. Н. Киселева. — Москва : ДМК Пресс, 2019. — 240 с. — ISBN 978-5-97060-739-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/123710 (дата обращения: 02.09.2021). — Режим доступа: для авториз. пользователей.
5. Рябов, В. А. Современные веб-технологии : учебное пособие / В. А. Рябов, А. И. Несвижский. — 2-е изд. — Москва : ИНТУИТ, 2016. — 1080 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/100499 (дата обращения: 08.12.2021). — Режим доступа: для авториз. пользователей.
6. Дешко, И. П. Управление сетевыми информационными системами: Курс лекций : учебное пособие / И. П. Дешко, К. Г. Кряженков. — Москва : РТУ МИРЭА, 2021. — 174 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/176536 (дата обращения: 08.12.2021). — Режим доступа: для авториз. пользователей.
7. Официальный сайт Spring [Электронный ресурс]: Режим доступа https://spring.io/projects/spring-framework свободный (дата обращения: 12.12.2021)
8. Статья на habr – «Redis — главное хранилище?» [Электронный ресурс]: Режим доступа https://habr.com/ru/post/178525/ свободный (дата обращения: 12.12.2021)
9. Статья на habr – «Архитектура Rest» [Электронный ресурс]: Режим доступа https://habr.com/ru/post/38730/ свободный (дата обращения: 10.12.2021)
10. Статья на javarush – «Что такое JSP? Разбираемся с возможностями на практике» [Электронный ресурс]: Режим доступа https://javarush.ru/groups/posts/2655-chto-takoe-jsp-razbiraemsja-s-vozmozhnostjami-na-praktike свободный (дата обращения: 11.12.2021)