# Введение

В бизнесе обеспечение эффективного и быстрого обслуживания клиентов в ресторанах является важной задачей. Один из способов улучшения качества обслуживания и автоматизации бизнес-процессов включает использование информационных систем, которые упрощают взаимодействие между администраторами и официантами. Для создания такой информационной системы было разработано клиент-серверное приложение

[1] для ресторана на основе архитектуры REST [7] и языка программирования Java. В рамках разработки был использован фреймворк Spring Boot [2] и ORM (Object-Relational Mapping) для связи Spring с СУБД. Приложение состоит из серверной и клиентской частей [1]. Серверная часть содержит контроллеры, сервисы и репозитории с логикой таблиц СУБД [3]. Клиентская часть представляет собой графический интерфейс пользователя, который обеспечивает удобное взаимодействие с приложением.

Для обеспечения удобства пользователей на стороне клиента были реализованы несколько окон с удобным дизайном и расположением элементов интерфейса [3]. В ходе разработки приложения уделено внимание возможным ошибкам, таким как ввод данных ошибочного типа, недопустимые значения, отсутствие требуемых данных и другие непредвиденные ситуации.

Выбранная тема курсовой работы имеет высокую значимость в настоящее время, так как программные решения играют важную роль в современном обществе, обеспечивая возможность передачи информации различных типов в режиме реального времени. Общение и взаимодействие являются ключевыми аспектами современного общества, включая сферу обслуживания ресторанов.

***Объектом исследования*** является клиент-серверное приложение для ресторана, а ***предметом исследования*** – разработка и реализация данной информационной системы для повышения качества обслуживания в ресторанах.

Таким образом, ***целью*** проекта является разработка клиент-серверного приложения для персонала ресторана с целью автоматизации бизнес-процесса.

Для достижения желаемого результата требуется выполнить следующие ***задачи***:

* разработка клиент-серверного приложения на языке программирования Java;
* создание серверной части приложения на основе фреймворка Spring Boot, отвечающей за обработку запросов клиентов;
* разработка клиентской части приложения с графическим интерфейсом с помощью HTML, CSS, и JavaScript, обеспечивающей удобный интерфейс для взаимодействия с сервером.

Для решения вышепоставленных задач используются инструментальные средаства, такие как Java, Spring Boot, MySQL, ORM .

В целях обеспечения удобного взаимодействия с пользователем, клиентская сторона приложения будет включать несколько окон с интуитивно понятным дизайном и оптимальным размещением элементов интерфейса.

На серверной стороне приложения будет использоваться для установления связи между фреймворком Spring и базой данных. Также будет применяться архитектурный шаблон MVC (Model-View-Controller), где контроллеры, сервисы и репозитории с логикой таблиц базы данных будут соответствующим образом разделены и организованы.

Для хранения данных используется реляционная база данных, которая содержит информацию о заказах, меню и пользователях системы.

Разработанное приложение должно обеспечивать следующие функции:

* + отображение данных предметной области в таблице;
  + добавление, удаление и редактирование объектов в БД;
  + фильтрация записей;
  + процесс получения информации из базы данных и представление статистических данных.

На стороне frontend требуется создать удобный и понятный графический интерфейс пользователя.

Разработка информационной системы ресторана включает в себя создание веб-сайта, предназначенного для автоматизации работы официантов и администраторов. Сайт предоставляет возможность администратору регистрировать новых официантов и новых посетителей ресторана. Официанты могут авторизоваться в свои учетные записи и просматривать информацию о клиентах, в том числе ФИО клиента, дату и время его прихода, номер столика и имя официанта, который обслуживает его. Администратор имеет больше прав, чем официанты, и может редактировать любую информацию в базе данных.

Благодаря использованию кроссплатформенных решений, клиентская часть приложения может быть запущена на любых устройствах, поддерживающих платформу Java, что обеспечивает удобство использования для пользователей. Пользовательский интерфейс приложения является интуитивно понятным и простым в использовании. Благодаря этому приложению, весь процесс управления рестораном становится более эффективным и удобным для пользователей.

Данное приложение также имеет потенциал для дальнейшего развития с добавлением новых функциональных возможностей, улучшением пользовательского интерфейса и оптимизацией процессов управления рестораном.

**Список тест-сьютов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Автор | Приоритет | Заголовок | Список тест-кейсов |
| **1** | user | 2 | Модуль пользователя | 1. Авторизация  2. Просмотр |
| **2** | admin | 1 | Модуль администратора | 1. Авторизация  2. Регистрация  3. Просмотр и редактирование посетителей |

**Список тест кейсов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Описание (Тип) | Предусловия | Шаги | Ожидаемый результат |
| 1 | Авторизация (позитивный) | 1. Пользователь находится на странице входа в личный кабинет  2. Пользователь был ранее зарегистрирован в системе | 1. Ввести в поля «Логин» и «Пароль» логин и пароль пользователя 2. Нажать кнопку «Войти» | 1. Пользователь перенаправляется на страницу личного кабинета |
| 2 | Регистрация нового пользователя (только для администраторов, позитивный) | Администратор находится на странице входа в личный кабинет | 1. Заполнены данные по новому пользователю  2. Выбрана роль нового пользователя (admin/user) | В базе данных создается новый пользователь с полными/ограниченными правами |
| 3 | Просмотр занятых столиков официантом, которые он не обслуживает (негативный) | 1. Пользователь находится на главной странице  2. Пользователь авторизован как user | Просмотр таблицы в верхней части главной страницы | Полный просмотр доступен только для пользователей с ролью admin |
| 4 | Регистрация пользователей  (негативный) | Новый незарегистрированный сотрудник | Существующий user доводит до администратора информацию о необходимости регистрации пользователя | Регистрация проводится только администраторами |
| 5 | Просмотр и редактирование посетителей  (позитивный) | Авторизированный администратор | 1. Admin заполняет данные о пользователе (посетитель)  2. Admin может редактировать информацию о любом посетителе | 1. Добавление и изменение информации о посетителях  2. Отображение данных в таблице и на графике на главной странице |

**Описание эквивалентных классов**

- Класс "Авторизация" - включает в себя тест-кейсы "Авторизация (позитивный)" и "Регистрация нового пользователя (только для администраторов, позитивный)". В данном классе проверяется возможность авторизации пользователя в системе, а также возможность регистрации нового пользователя администратором.

- Класс "Просмотр и редактирование посетителей" - включает в себя тест-кейсы "Просмотр занятых столиков официантом, которые он не обслуживает (негативный)" и "Просмотр и редактирование посетителей (позитивный)". В данном классе проверяется возможность просмотра и редактирования информации о посетителях ресторана.

**Расчет количества тестов**

Для класса "Авторизация" требуется провести 2 теста. Один тест для проверки возможности авторизации пользователя в системе, а другой - для проверки возможности регистрации нового пользователя администратором.

Для класса "Просмотр и редактирование посетителей" требуется провести 2 теста. Один тест для проверки возможности просмотра занятых столиков официантом, которые он не обслуживает, а другой - для проверки возможности просмотра и редактирования информации о посетителях ресторана.

Итого, требуется провести 4 теста.

**Ручное тестирование**

| **ID** | **Описание теста** | **Предусловия** | **Шаги** | **Ожидаемый результат** |
| --- | --- | --- | --- | --- |
| 1 | Авторизация (позитивный) | 1. Пользователь находится на странице входа в личный кабинет | 1. Ввести в поля "Логин" и "Пароль" логин и пароль пользователя | 1. Пользователь перенаправляется на страницу личного кабинета |
| 2 | Регистрация нового пользователя (только для администраторов, позитивный) | Администратор находится на странице входа в личный кабинет | 1. Заполнить данные по новому пользователю 2. Выбрать роль нового пользователя (admin/user) | В базе данных создается новый пользователь с полными/ограниченными правами |
| 3 | Просмотр занятых столиков официантом, которые он не обслуживает (негативный) | 1. Пользователь находится на главной странице 2. Пользователь авторизован как user | Просмотр таблицы в верхней части главной страницы | Полный просмотр доступен только для пользователей с ролью admin |
| 4 | Регистрация пользователей (негативный) | Новый незарегистрированный сотрудник | Существующий user доводит до администратора информацию о необходимости регистрации пользователя | Регистрация проводится только администраторами |
| 5 | Просмотр и редактирование посетителей (позитивный) | Авторизированный администратор | 1. Admin заполняет данные о пользователе (посетитель) 2. Admin может редактировать информацию о любом посетителе | Добавление и изменение информации о посетителях; Отображение данных в таблице и на графике на главной странице |

|  |  |  |
| --- | --- | --- |
| Дата и время проведения теста | ID тест-кейса | Результат |
| 23.04.2024 16:09 | 1 | Успешно |
| 23.04.2024 16:15 | 2 | Успешно |
| 23.04.2024 16:19 | 3 | Успешно |
| 23.04.2024 16:24 | 4 | Успешно |
| 23.04.2024 16:30 | 5 | Успешно |

Список выявленных дефектов: дефекты не выявлены

**Задание к лабораторной 02.05**

a. Цель работы:

Цель данной лабораторной работы состоит в проведении анализа документации, описывающей требования к разработке программного обеспечения, с целью оценки ее качества и выявления несоответствий

b. Описание тестируемой документации:

Для анализа была выбрана техническая документация к проекту "Разработка клиент-серверного приложения для автоматизации работы ресторана". Эта документация описывает требования к функциональности приложения, его архитектуре, структуре базы данных и интерфейсу пользователя.

c. Описание критериев качества тестируемой документации:

Критерии качества для данной документации включают полноту, однозначность, непротиворечивость, актуальность, структурированность и тестируемость.

d. Описание и обоснование метода тестирования документации:

Для тестирования документации будет использован метод рецензирования. Этот метод позволяет провести анализ документов разными специалистами, выявить проблемы и недочеты с различных точек зрения, и предложить рекомендации по их устранению.

e. Список несоответствий в документации критериям качества:

1. Полнота: Не вся функциональность описана в документации, например, отсутствует описание процесса регистрации новых посетителей ресторана.

2. Однозначность: Некоторые термины и аббревиатуры не расшифрованы, что может привести к недопониманию.

3. Непротиворечивость: В разных разделах документации противоречиво описывается процесс регистрации новых пользователей.

4. Актуальность: Некоторая информация в документации устарела и не соответствует реальной реализации проекта.

5. Структурированность: Структура документации не всегда удобна для поиска нужной информации, например, некоторые разделы расположены в неудобном порядке.

6. Тестируемость: Некоторые функциональные требования не имеют четких критериев проверки, что затрудняет тестирование.

f. Выводы по работе:

Анализ документации показал, что она нуждается в доработке и улучшении с точки зрения всех критериев качества. Рекомендуется провести пересмотр и дополнение документов в соответствии с выявленными несоответствиями.

16.05 Лаба 2 во 2 аттестации

c. Список видов, типов, методов и уровней тестирования для выбранного проекта

Виды тестирования

1. Функциональное тестирование

- Проверка соответствия требований функциональной спецификации.

- Примеры: Авторизация, регистрация новых пользователей, просмотр и редактирование данных.

2. Нагрузочное тестирование

- Проверка производительности при увеличении нагрузки.

- Примеры: Тестирование реакции системы на большое количество запросов.

3. Стресс-тестирование

- Проверка функционирования системы в условиях экстремальных нагрузок.

- Примеры: Проверка устойчивости системы при одновременном входе большого количества пользователей.

4. Юзабилити тестирование

- Проверка удобства использования интерфейса.

- Примеры: Тестирование простоты навигации и логичности расположения элементов интерфейса.

5. Тестирование безопасности

- Проверка системы на уязвимости.

- Примеры: Тестирование защиты данных пользователей от несанкционированного доступа.

6. Тестирование совместимости

- Проверка работы на разных платформах и браузерах.

- Примеры: Проверка работы приложения на различных операционных системах и браузерах.

7. Тестирование конфигурации

- Проверка работы системы на различных аппаратных платформах.

- Примеры: Тестирование работы с разными конфигурациями серверного оборудования.

8. Тестирование локализации

- Проверка правильности перевода и отображения интерфейса на разных языках.

- Примеры: Проверка корректности перевода сообщений и интерфейсов.

Типы тестирования

1. Позитивное тестирование

- Тестирование на данных, которые соответствуют нормальному поведению системы.

- Примеры: Ввод корректных данных для авторизации.

2. Негативное тестирование

- Тестирование на данных, которые соответствуют нештатному поведению системы.

- Примеры: Ввод некорректных данных для регистрации.

Методы тестирования

1. Ручное тестирование

- Проверка функциональности приложения вручную.

- Примеры: Проверка работы форм ввода и вывода данных.

2. Автоматизированное тестирование

- Использование скриптов и программ для автоматической проверки.

- Примеры: Использование Selenium для тестирования интерфейса.

3. Статическое тестирование

- Анализ кода и документации без выполнения программы.

- Примеры: Рецензирование кода и требований.

4. Динамическое тестирование

- Проверка функциональности с выполнением программы.

- Примеры: Запуск приложения и проверка его работы с реальными данными.

Уровни тестирования

1. Модульное тестирование

- Проверка отдельных функций или модулей.

- Примеры: Тестирование функции авторизации пользователя.

2. Интеграционное тестирование

- Проверка взаимодействия между модулями.

- Примеры: Тестирование связи между модулем пользователя и базой данных.

3. Системное тестирование

- Проверка всей системы в целом.

- Примеры: Проверка полного рабочего цикла от авторизации до завершения заказа.

4. Приемочное тестирование

- Проверка соответствия системы требованиям перед выпуском.

- Примеры: Финальная проверка перед развертыванием системы на сервере клиента.

d. Примеры дефектов, характерных для каждого вида тестирования

1. Функциональное тестирование

- Примеры дефектов:

- Неправильная авторизация пользователя.

- Ошибки при регистрации новых пользователей.

- Некорректное отображение данных о посетителях.

2. Нагрузочное тестирование

- Примеры дефектов:

- Замедление работы системы при увеличении числа запросов.

- Потеря данных при большой нагрузке.

3. Стресс-тестирование

- Примеры дефектов:

- Система перестает отвечать при экстремально высокой нагрузке.

- Непредсказуемое поведение системы при исчерпании ресурсов.

4. Юзабилити тестирование

- Примеры дефектов:

- Трудности в навигации по интерфейсу.

- Некорректное отображение элементов интерфейса на разных устройствах.

5. Тестирование безопасности

- Примеры дефектов:

- Уязвимость к SQL-инъекциям.

- Недостаточная защита паролей пользователей.

6. Тестирование совместимости

- Примеры дефектов:

- Некорректное отображение интерфейса в разных браузерах.

- Проблемы с функциональностью на разных операционных системах.

7. Тестирование конфигурации

- Примеры дефектов:

- Сбой системы при работе с определенным оборудованием.

- Некорректная работа с различными конфигурациями серверов.

8. Тестирование локализации

- Примеры дефектов:

- Неправильный перевод интерфейса.

- Ошибки в отображении локализованных данных.

9. Позитивное тестирование

- Примеры дефектов:

- Неожиданные ошибки при корректных данных.

10. Негативное тестирование

- Примеры дефектов:

- Система не обрабатывает некорректные данные должным образом.

- Ошибки при вводе невалидных данных.

**Заключение**

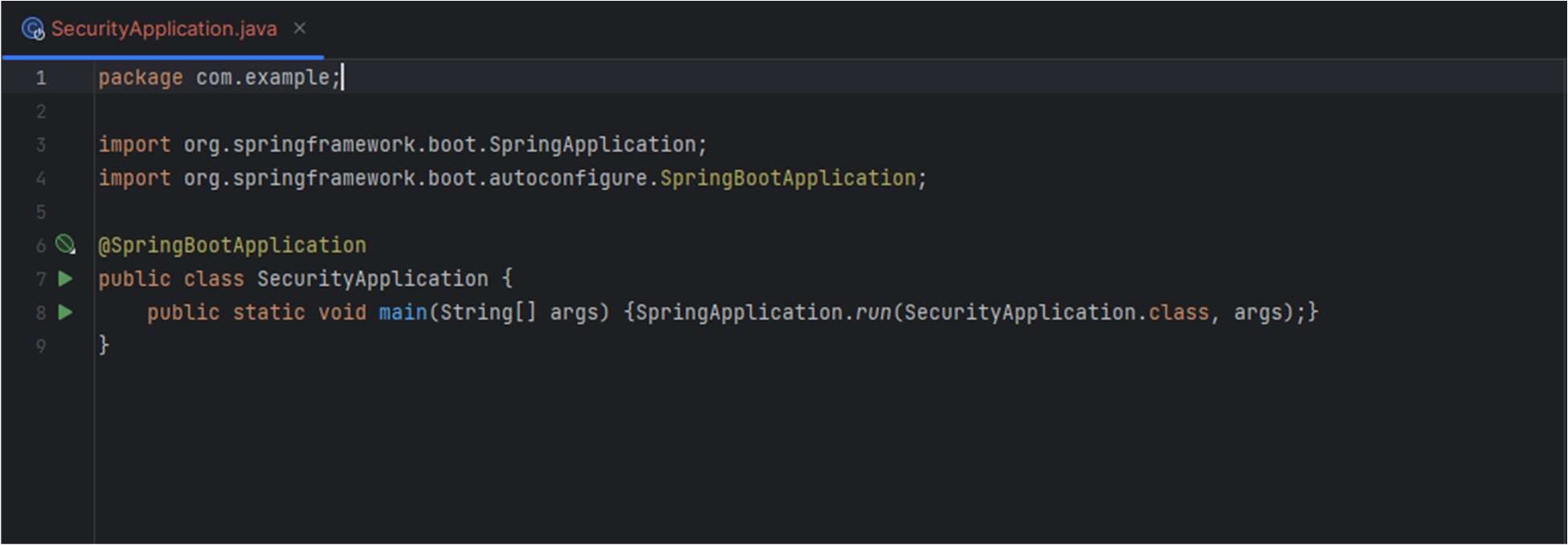
В результате выполнения проекта было успешно создано приложение для ресторана, которое позволяет упростить процесс управления рестораном и облегчить работу персонала. Приложение обладает высокой стабильностью на различных платформах и не вызывает проблем при эксплуатации. Разработан клиент-серверный интерфейс для взаимодействия с приложением, который позволяет официантам ресторана быстро и удобно отслеживать информацию о клиентах. Приложение имеет простой и интуитивно понятный пользовательский интерфейс, что позволяет пользователям без труда пользоваться им и не требует большого времени на обучение. Была разработана база данных, которая позволяет сохранять информацию о посетителях ресторана, что делает процесс управления рестораном более прозрачным и эффективным. В ходе работы над проектом были получены навыки в области разработки сложных приложений, управления базами данных и пользовательскими интерфейсами. Обретен ценный опыт, который может быть полезно применен для прогресса проекта и расширения его возможностей.

**Список использованных источников**

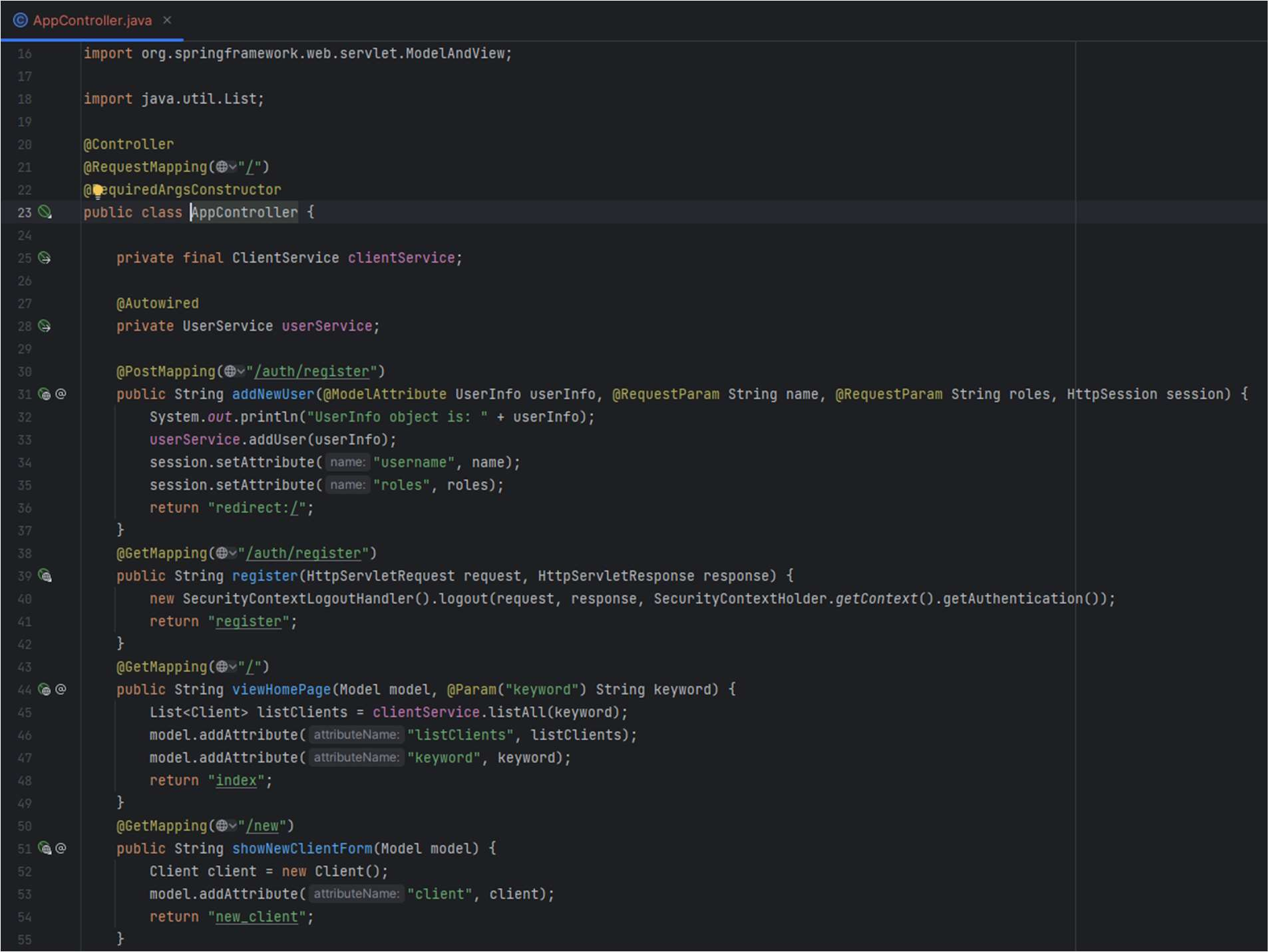
1. Коузен К. Современный Java. Рецепты программирования. - М.: ДМК Пресс, 2018. - 274 с.
2. Козмина Ю., Харроп Р. Spring 5 для профессионалов. - Киев: Диалектика-Вильямс, 2019. - 1120 с.
3. Мартин Роберт К. Чистый код. Создание анализ и рефакторинг. - СПб: Питер, 2019. - 464 с.
4. Связанные сущности в Hibernate [Электронный ресурс] // java-online. URL: <http://java-online.ru/hibernate-entities.xhtml>
5. Java json parser – пример работы парсера [Электронный ресурс] // hr- vector. URL: https://hr-vector.com/java/json-parser
6. Сериализация и десереализация в Java [Электронный ресурс] // javarush. URL: https://javarush.ru/groups/posts/2022-serializacija-i-deserializacija- v-java
7. REST with Spring Tutorial [Электронный ресурс] // baeldung. URL: https://[www.baeldung.com/rest-with-spring-series](http://www.baeldung.com/rest-with-spring-series)

**Приложения**

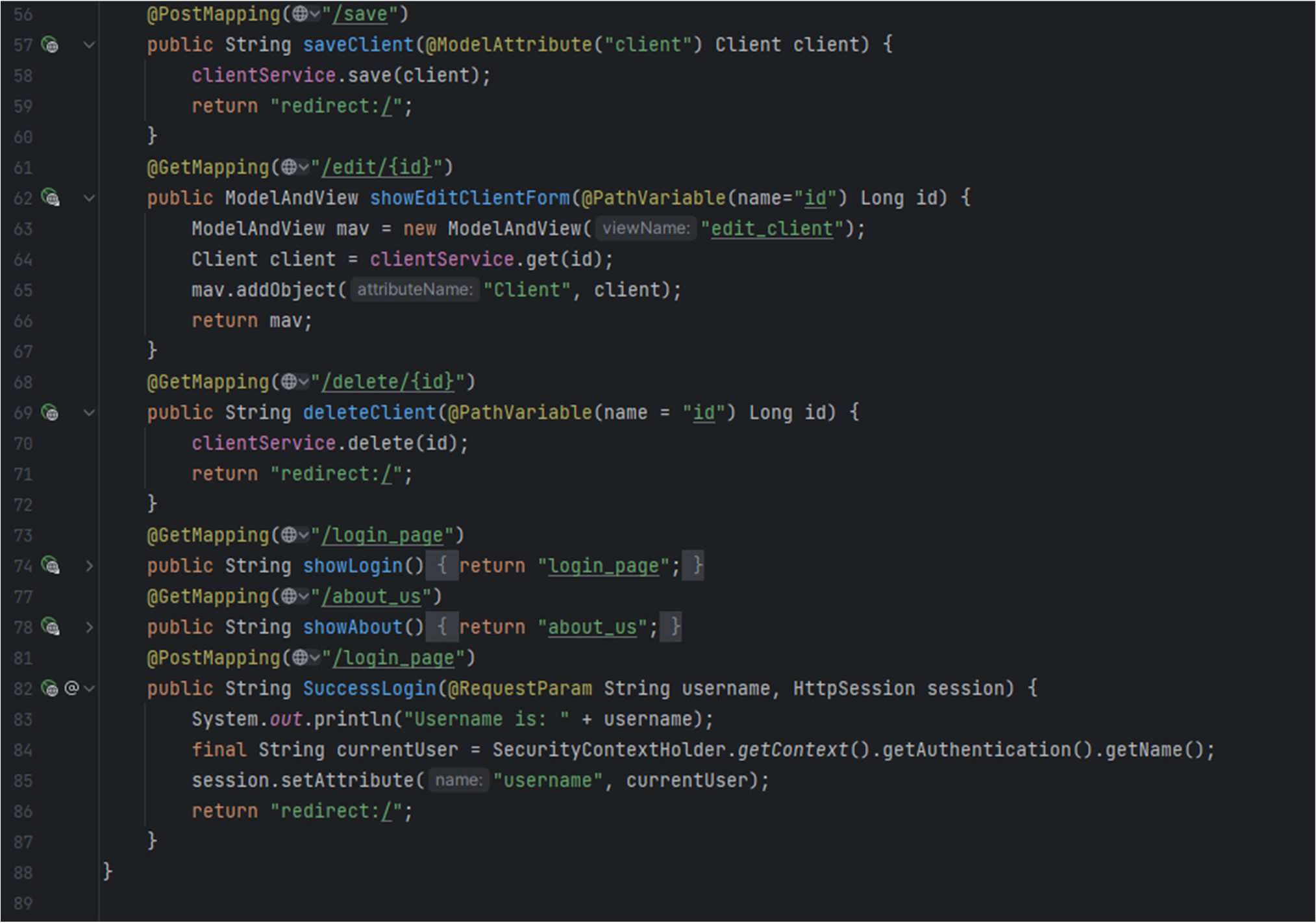
Приложение А – Фрагмент исходного кода «Запуск программы»



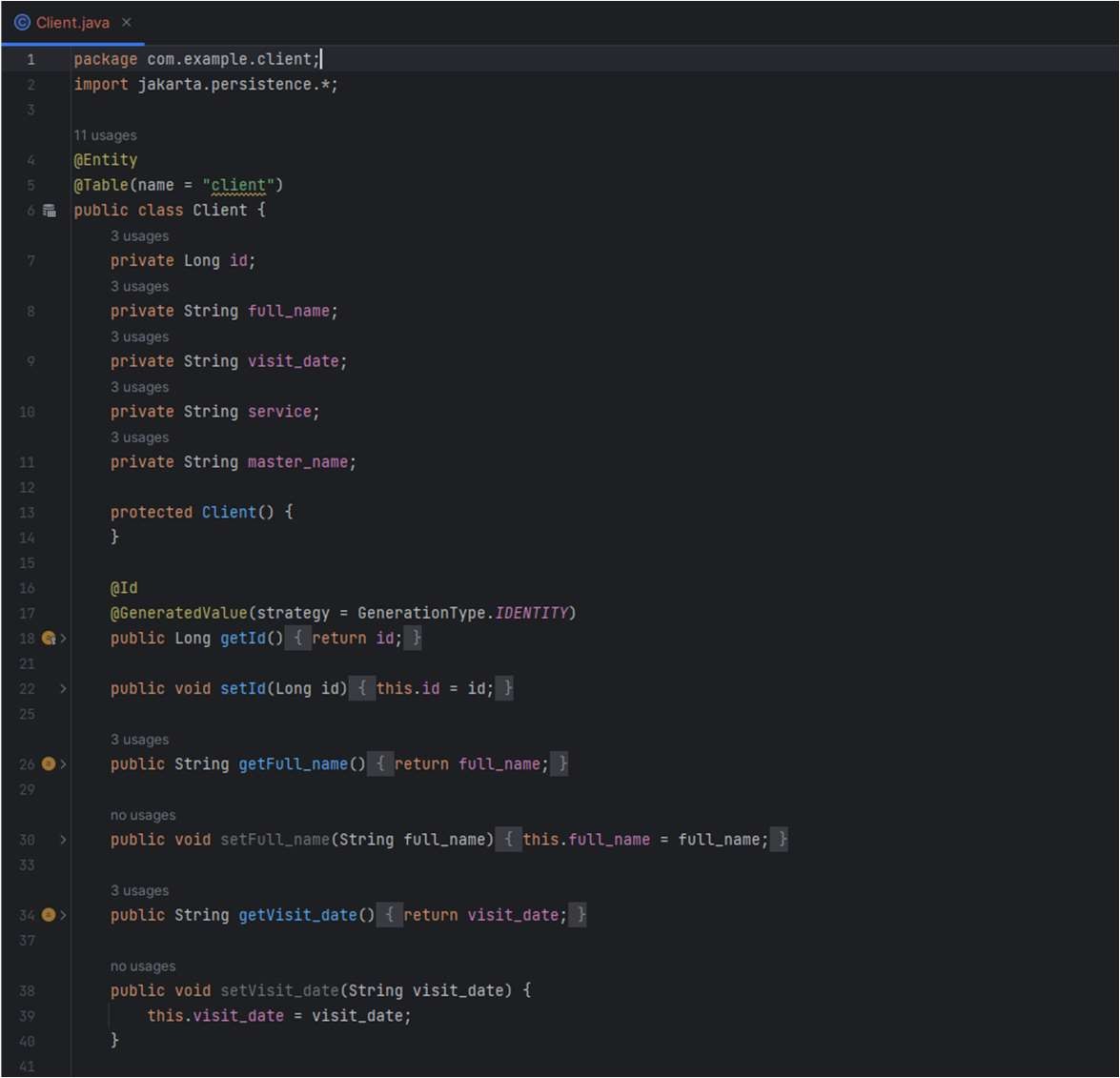
Приложение Б – Фрагмент исходного кода «Контроллер для обработки запросов»



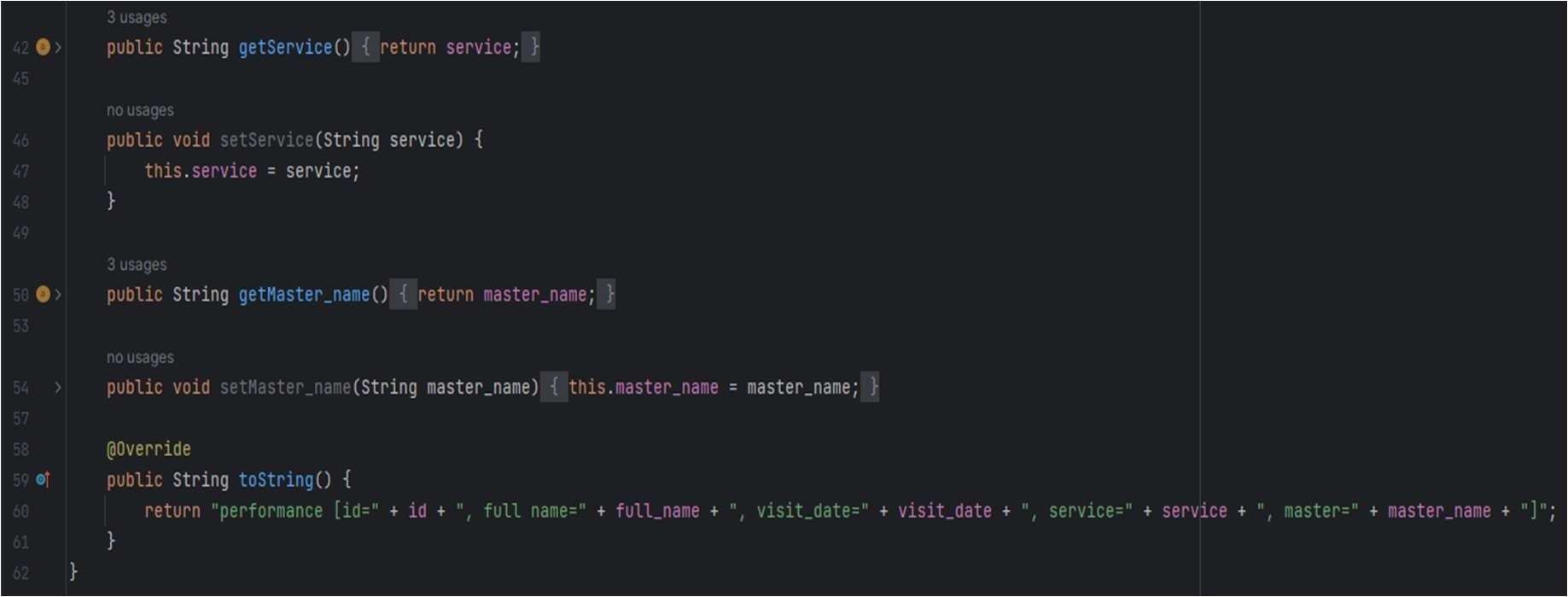
Приложение Б – Фрагмент исходного кода «Контроллер для обработки запросов» (Продолжение)



Приложение В – Фрагмент исходного кода «класс Client»



Приложение В – Фрагмент исходного кода «класс Client» (Продолжение)



Приложение Г – Фрагмент исходного кода «репозиторий для работы с данными сущности Client»



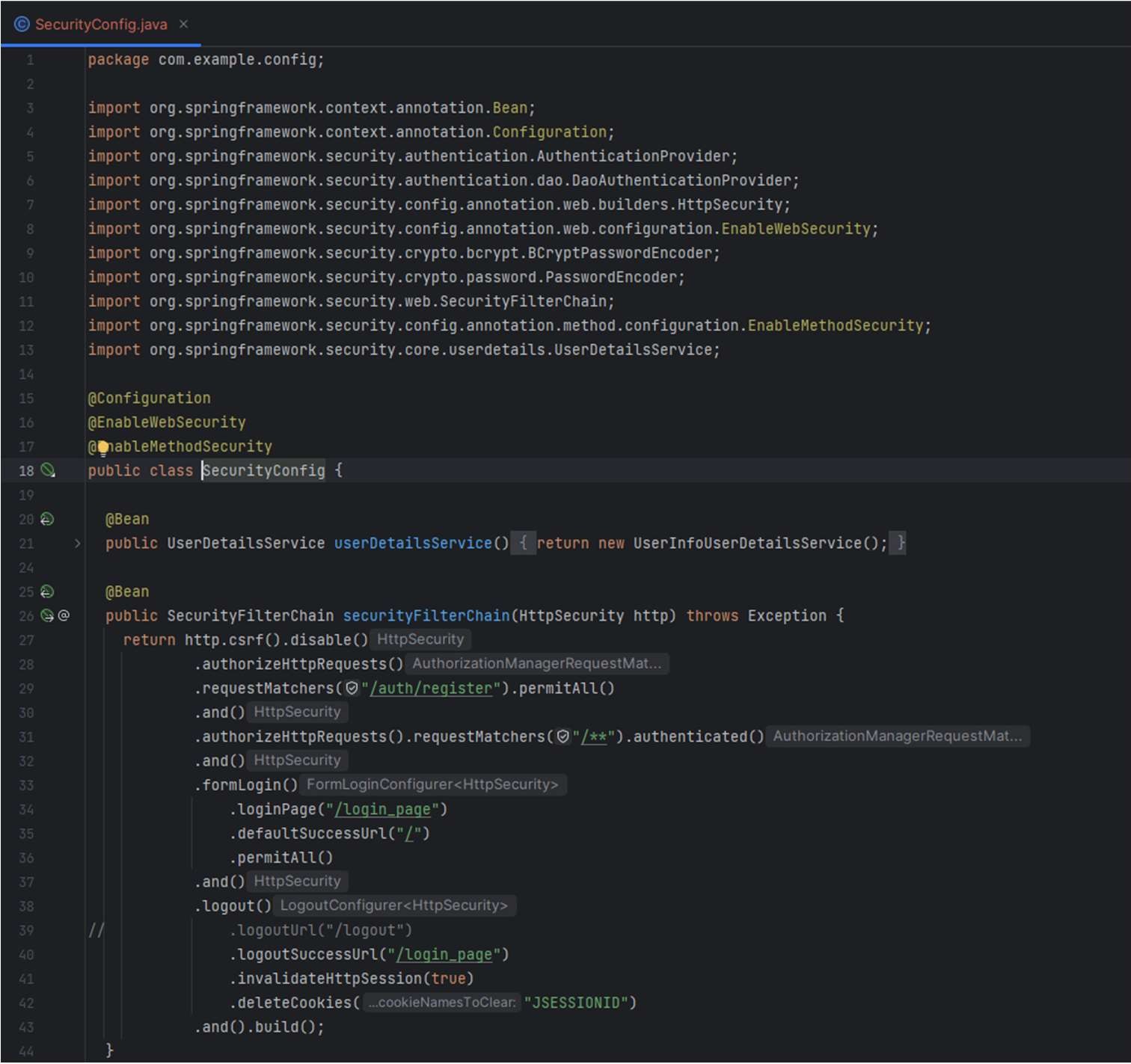
Приложение Д – Фрагмент исходного кода «сервисы, связанные с сущностью Client и репозиторием»



Приложение Е – Фрагмент исходного кода «конфигурация Spring MVC»



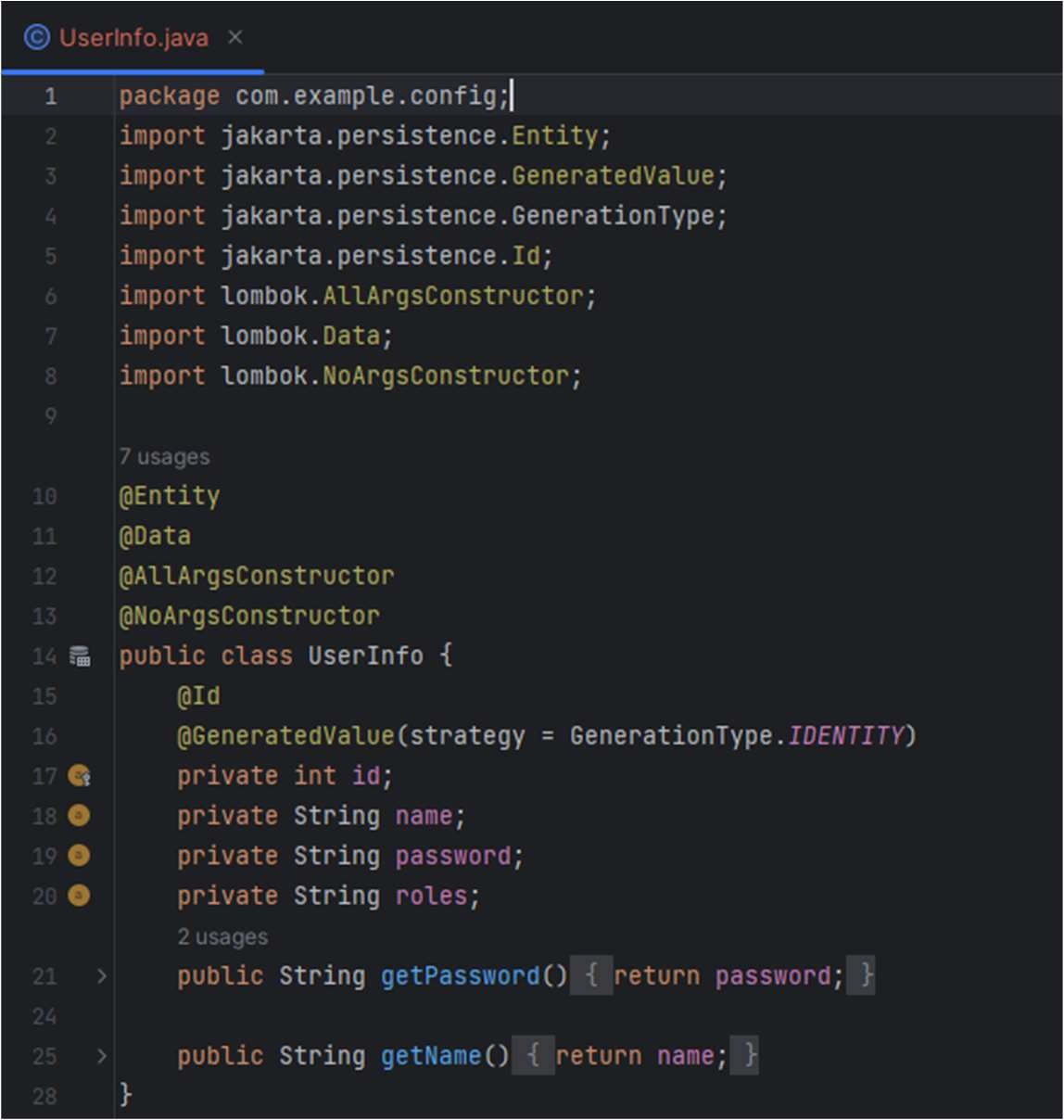
Приложение Ж – Фрагмент исходного кода «конфигурация Spring Security»



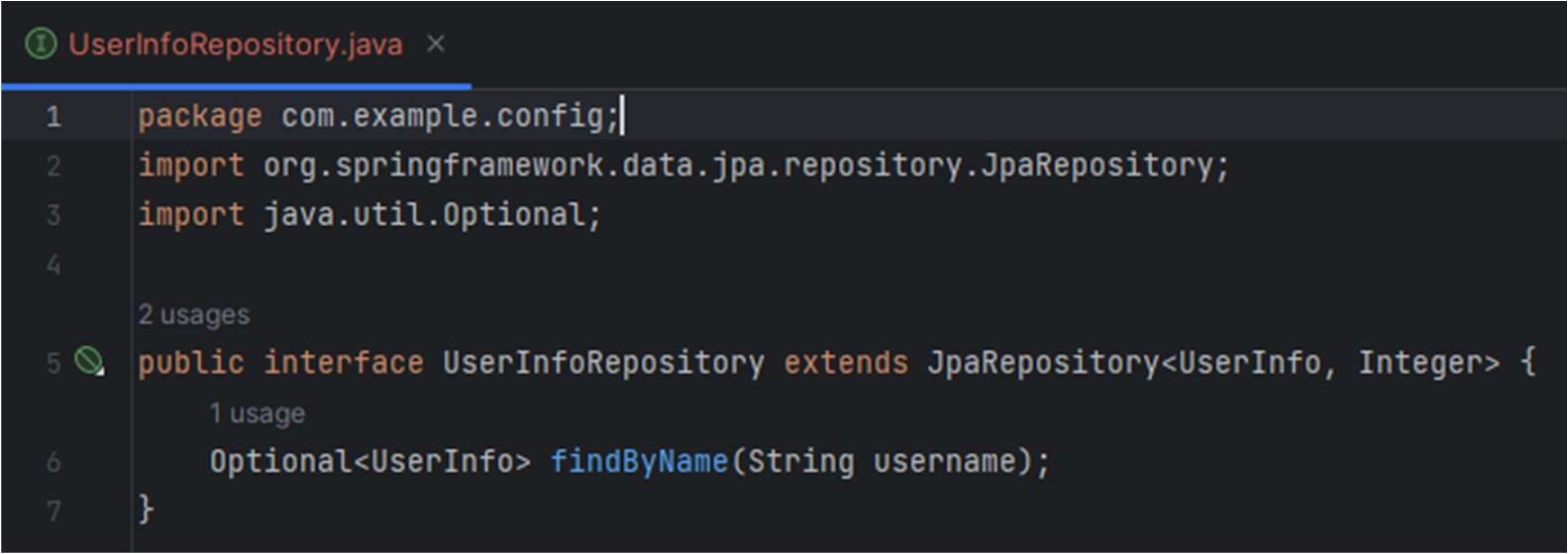
Приложение Ж – Фрагмент исходного кода «конфигурация Spring Security» (Продолжение)



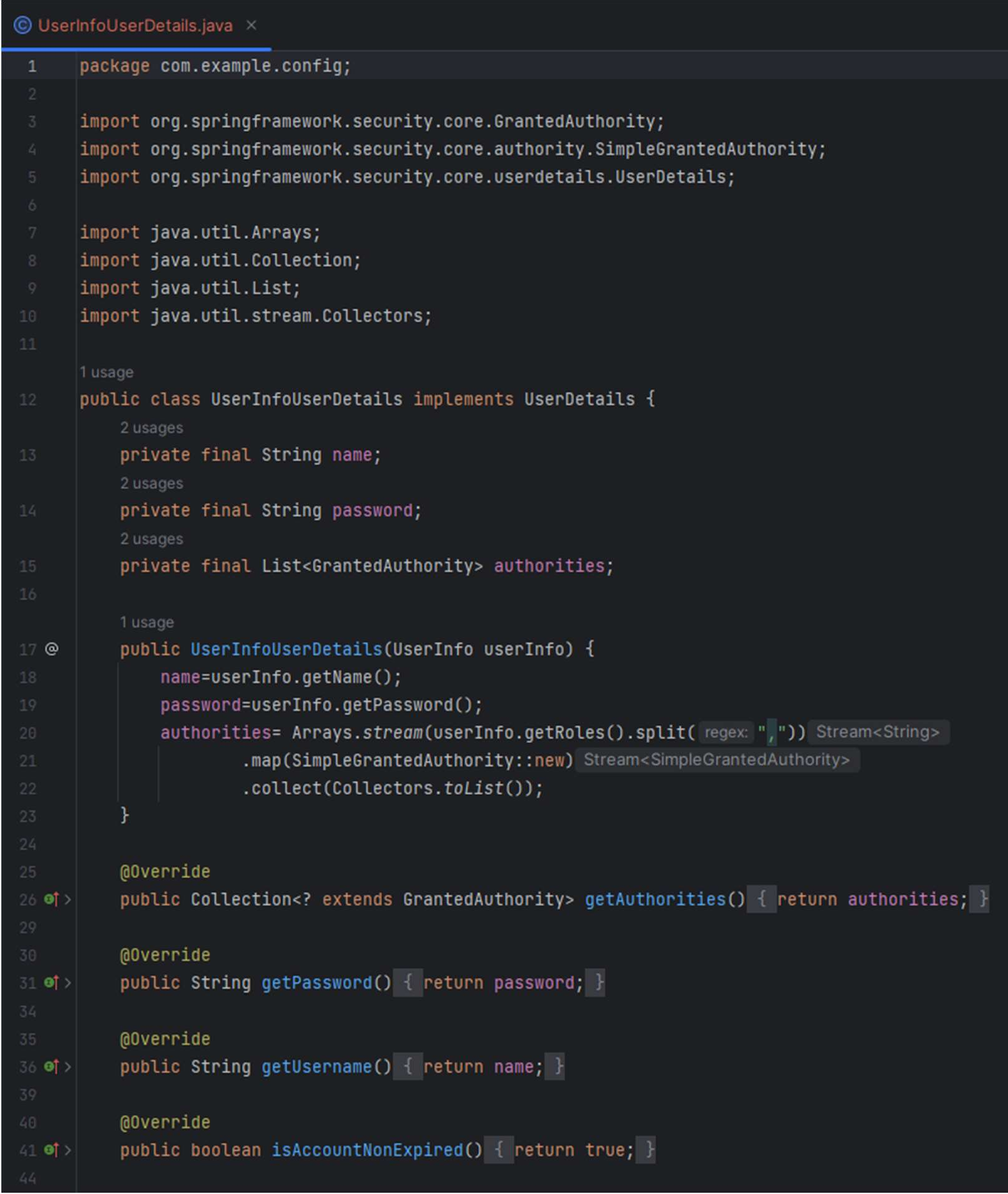
Приложение З – Фрагмент исходного кода «модель информации о пользователе»



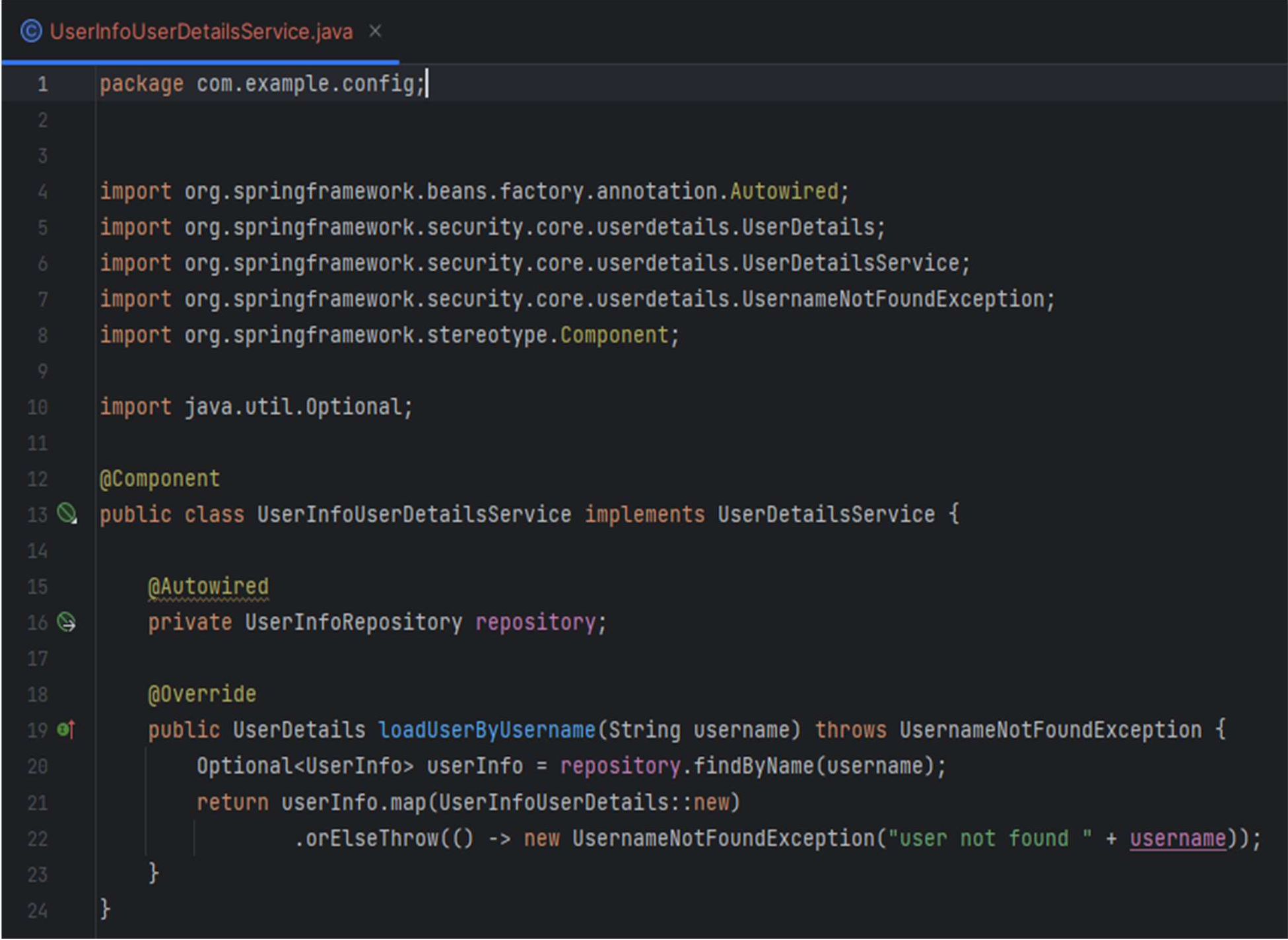
Приложение И – Фрагмент исходного кода «репозиторий для работы с данными сущности User»



Приложение К – Фрагмент исходного кода «аутентификации пользователя в Spring Security»



Приложение Л – Фрагмент исходного кода «аутентификация пользователей в системе»



Приложение М – Фрагмент исходного кода «сервис для добавления новых пользователей в репозиторий»

