Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ярославский государственный технический университет»

Кафедра «Цифровых систем»

Курсовая работы защищена

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель,

ассистент

\_\_\_\_\_\_\_\_\_\_\_\_\_ Пашичев В.С

«\_\_»\_\_\_\_\_\_\_\_\_\_\_2024

**РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ НА SPRING**

Расчетно-пояснительная записка к курсовой работе по дисциплине

«Технологии программирования»

ЯГТУ 09.03.02 – 000 КП

|  |  |
| --- | --- |
| Нормоконтролер  Ассистен  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Пашичев В.С.  «\_\_»\_\_\_\_\_\_\_\_\_\_\_2024 | Проект выполнил  студент группы ЦИС-27  Бычков Е.А\_\_\_\_\_\_\_\_  «\_\_»\_\_\_\_\_\_\_\_\_\_\_2024 |

2024

Введение

Современные информационные технологии развиваются стремительными темпами, и с каждым годом растет потребность в высокопроизводительных и надежных компьютерных системах. В связи с этим задача подбора оптимальной конфигурации компьютера становится все более актуальной как для индивидуальных пользователей, так и для корпоративного сектора. Правильный выбор компонентов компьютера позволяет не только сэкономить средства, но и обеспечить высокую производительность и долговечность системы.

Целью данной курсовой работы является разработка веб-приложения для подбора конфигурации компьютера с использованием Java Spring. Веб-приложение позволит пользователям легко и быстро сформировать оптимальную конфигурацию компьютера, исходя из их индивидуальных потребностей и бюджета.

Использование Java Spring в разработке данного веб-приложения обусловлено его широкими возможностями и преимуществами, такими как высокая производительность, масштабируемость и гибкость. Spring Framework предоставляет богатый набор инструментов для создания надежных и устойчивых веб-приложений, что делает его идеальным выбором для реализации проекта данного типа.

В ходе работы будет рассмотрено архитектурное решение проекта, описаны основные этапы разработки и тестирования, а также проведен анализ результатов и перспективы дальнейшего развития веб-приложения.

Общая диаграмма классов представлена на Рисунке 1:

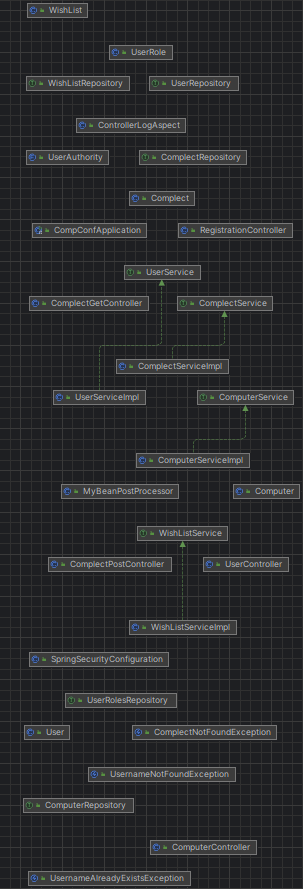


Рисунок – Диаграмма классов в IDE

**Проектирование базы данных**

Проектирование базы данных включает в себя разработку логической и физической модели данных.

**Логическая модель**

На логическом уровне определяются сущности, их атрибуты и взаимосвязи между ними. На основе предоставленной схемы определены следующие сущности:

* **User**
  + id (PRIMARY KEY)
  + login
  + password
* **WishList**
  + id (PRIMARY KEY)
  + idUser (FOREIGN KEY)
* **Computer**
  + id (PRIMARY KEY)
  + idWishList (FOREIGN KEY)
  + idCPU (FOREIGN KEY)
  + idGPU (FOREIGN KEY)
  + idMotherBoard (FOREIGN KEY)
  + idCooler (FOREIGN KEY)
  + idRAM (FOREIGN KEY)
  + countRAM
  + idSSD (FOREIGN KEY)
  + countSSD
  + idHDD (FOREIGN KEY)
  + countHDD
  + idComputerCase (FOREIGN KEY)
  + idPowerUnit (FOREIGN KEY)
  + totalCost
* **Complect**
  + id (PRIMARY KEY)
  + type
  + name
  + cost
  + description

Для проектирования была использована СУБД PostgreSQL

Нормализация базы данных выполняется для устранения избыточности данных и обеспечения целостности. В данном случае схема соответствует третьей нормальной форме (3NF), так как:

* Все атрибуты атомарны.
* Все неключевые атрибуты зависят от первичного ключа.
* Нет транзитивных зависимостей.

Класс Complect

Класс Complect представляет сущность "Комплектующий" в системе подбора конфигурации компьютера. Он описывает отдельный компонент компьютера, такой как процессор, видеокарта, материнская плата и другие комплектующие. Класс содержит основные атрибуты комплектующего и взаимосвязи с другими классами, что позволяет реализовать необходимые функции для работы веб-приложения.

**Импортируемые библиотеки и аннотации**

* com.fasterxml.jackson.annotation.JsonIgnore и com.fasterxml.jackson.annotation.JsonManagedReference: Используются для управления сериализацией JSON, чтобы избежать циклических зависимостей и рекурсивных ссылок.
* jakarta.persistence.\*: Набор аннотаций JPA (Java Persistence API) для маппинга класса на таблицу базы данных и указания различных аспектов поведения полей класса.
* lombok.\*: Библиотека Lombok для автоматической генерации стандартных методов, таких как геттеры, сеттеры, конструкторы и другие.

**Аннотации на уровне класса**

* @AllArgsConstructor, @NoArgsConstructor: Lombok-аннотации для автоматического создания конструктора с аргументами и конструктора без аргументов соответственно.
* @Getter, @Setter: Lombok-аннотации для генерации геттеров и сеттеров для всех полей класса.
* @Entity(name = "Complect"): Указывает, что класс является сущностью JPA и должен быть маппирован на таблицу Complect в базе данных.
* @Table(name = "Complect"): Задает имя таблицы в базе данных, с которой ассоциируется данный класс.

**Поля класса**

* id: Уникальный идентификатор комплектующего, автоматически генерируемый с использованием последовательности.
* type: Тип комплектующего (например, "CPU", "GPU").
* name: Название комплектующего.
* cost: Стоимость комплектующего.
* description: Описание комплектующего.

Класс Computer

Класс Computer представляет сущность "Компьютер" в системе подбора конфигурации компьютера. Он описывает конфигурацию компьютера, включающую различные комплектующие, такие как процессор, видеокарта, оперативная память и другие компоненты. Класс содержит атрибуты, соответствующие этим компонентам, и обеспечивает их взаимосвязь с другими сущностями, такими как WishList и Complect.

**Импортируемые библиотеки и аннотации**

* com.fasterxml.jackson.annotation.JsonBackReference: Используется для управления сериализацией JSON и предотвращения циклических зависимостей.
* jakarta.persistence.\*: Набор аннотаций JPA (Java Persistence API) для маппинга класса на таблицу базы данных и указания различных аспектов поведения полей класса.
* lombok.\*: Библиотека Lombok для автоматической генерации стандартных методов, таких как геттеры, сеттеры, конструкторы и другие.

**Аннотации на уровне класса**

* @AllArgsConstructor, @NoArgsConstructor: Lombok-аннотации для автоматического создания конструктора с аргументами и конструктора без аргументов соответственно.
* @Getter, @Setter: Lombok-аннотации для генерации геттеров и сеттеров для всех полей класса.
* @Entity(name = "Computer"): Указывает, что класс является сущностью JPA и должен быть маппирован на таблицу Computer в базе данных.
* @Table(name = "Computer"): Задает имя таблицы в базе данных, с которой ассоциируется данный класс.

**Поля класса**

* id: Уникальный идентификатор конфигурации компьютера, автоматически генерируемый с использованием последовательности.
* wishList: Ссылка на сущность WishList, к которой относится данная конфигурация.
* cpu, gpu, motherBoard, cooler, ram, ssd, hdd, computerCase, powerUnit: Ссылки на сущность Complect, представляющие различные компоненты компьютера.
* countRAM, CountSSD, CountHDD: Количество соответствующих компонентов в конфигурации.
* totalCost: Общая стоимость конфигурации компьютера.

**Взаимосвязи с другими сущностями**

Каждое поле, представляющее компонент компьютера, аннотировано @ManyToOne, что указывает на связь "многие-к-одному" с сущностью Complect. Это означает, что множество конфигураций могут содержать один и тот же компонент.

* fetch = FetchType.EAGER: Определяет стратегию загрузки данных. FetchType.EAGER означает, что связанные данные будут загружены вместе с основной сущностью.
* @JoinColumn(name = "column\_name"): Указывает имя колонки в таблице базы данных, которая будет использоваться для хранения внешнего ключа.

Класс User

Класс User представляет пользователя в системе подбора конфигурации компьютера. Этот класс реализует интерфейс UserDetails из Spring Security, что позволяет использовать его для аутентификации и авторизации в приложении. Класс включает основные атрибуты пользователя, методы, необходимые для интеграции с Spring Security, и связи с другими сущностями, такими как UserRole и WishList.

**Импортируемые библиотеки и аннотации**

* com.fasterxml.jackson.annotation.JsonIgnore: Используется для исключения полей из сериализации JSON, чтобы избежать рекурсивных ссылок и утечки чувствительных данных.
* jakarta.persistence.\*: Набор аннотаций JPA (Java Persistence API) для маппинга класса на таблицу базы данных и указания различных аспектов поведения полей класса.
* lombok.\*: Библиотека Lombok для автоматической генерации стандартных методов, таких как геттеры, сеттеры, конструкторы и другие.
* org.springframework.security.core.GrantedAuthority, org.springframework.security.core.userdetails.UserDetails: Интерфейсы Spring Security для определения пользовательских данных, необходимых для аутентификации и авторизации.

**Аннотации на уровне класса**

* @AllArgsConstructor, @NoArgsConstructor: Lombok-аннотации для автоматического создания конструктора с аргументами и конструктора без аргументов соответственно.
* @Getter, @Setter: Lombok-аннотации для генерации геттеров и сеттеров для всех полей класса.
* @ToString: Lombok-аннотация для автоматического создания метода toString().
* @Accessors(chain = true): Lombok-аннотация для включения методов сеттера, возвращающих this, что позволяет использовать цепочку вызовов.
* @Entity(name = "App\_User"): Указывает, что класс является сущностью JPA и должен быть маппирован на таблицу App\_User в базе данных.
* @Table(name = "App\_User"): Задает имя таблицы в базе данных, с которой ассоциируется данный класс.

**Поля класса**

* id: Уникальный идентификатор пользователя, автоматически генерируемый с использованием последовательности.
* username: Имя пользователя.
* password: Пароль пользователя.
* expired: Флаг, указывающий, истек ли срок действия учетной записи.
* locked: Флаг, указывающий, заблокирована ли учетная запись.
* enabled: Флаг, указывающий, включена ли учетная запись.

**Взаимосвязи с другими сущностями**

* userRoles: Список ролей пользователя, аннотированный @OneToMany. Связь указывает, что один пользователь может иметь несколько ролей.
* wishList: Связь с сущностью WishList, представляющей список желаемых конфигураций пользователя. Аннотация @OneToOne указывает, что один пользователь может иметь только один список желаемых конфигураций.

**Реализация интерфейса UserDetails**

Класс User реализует методы интерфейса UserDetails, которые необходимы для интеграции с Spring Security:

* getAuthorities(): Возвращает коллекцию прав (ролей) пользователя.
* getPassword(): Возвращает пароль пользователя.
* getUsername(): Возвращает имя пользователя.
* isAccountNonExpired(), isAccountNonLocked(), isCredentialsNonExpired(), isEnabled(): Методы, возвращающие статус учетной записи пользователя (активна/неактивна).

Класс UserAuthority:

Класс UserAuthority представляет собой перечисление (enum), которое реализует интерфейс GrantedAuthority из Spring Security. Этот класс используется для определения прав (авторизаций), которые могут быть присвоены пользователю в системе. Права определяют, какие действия пользователь может выполнять в системе, такие как размещение заказов, управление заказами и полные права.

**Импортируемые библиотеки и интерфейсы**

* org.springframework.security.core.GrantedAuthority: Интерфейс Spring Security, который представляет собой право или авторизацию, присваиваемую пользователю.

**Перечисление прав**

Перечисление UserAuthority содержит три значения, каждое из которых представляет определенный уровень доступа:

* PLACE\_ORDERS: Право на размещение заказов.
* MANAGE\_ORDERS: Право на управление заказами.
* FULL: Полные права, которые могут включать в себя все доступные действия.

**Реализация метода getAuthority**

Метод getAuthority из интерфейса GrantedAuthority возвращает строковое представление права. В данном случае он возвращает имя константы перечисления, которое совпадает с ее названием.

Класс UserRole

Класс UserRole представляет роль пользователя в системе подбора конфигурации компьютера. Он содержит информацию о конкретном праве (авторизации), которое присвоено пользователю. Класс включает связи с сущностями User и UserAuthority для реализации механизма ролей и авторизаций.

**Импортируемые библиотеки и аннотации**

* com.fasterxml.jackson.annotation.JsonIgnore: Используется для исключения полей из сериализации JSON, чтобы избежать рекурсивных ссылок и утечки чувствительных данных.
* jakarta.persistence.\*: Набор аннотаций JPA (Java Persistence API) для маппинга класса на таблицу базы данных и указания различных аспектов поведения полей класса.
* lombok.\*: Библиотека Lombok для автоматической генерации стандартных методов, таких как геттеры, сеттеры, конструкторы и другие.

**Аннотации на уровне класса**

* @Data: Lombok-аннотация для генерации всех стандартных методов, таких как геттеры, сеттеры, toString(), equals() и hashCode().
* @Table(name = "user\_roles"): Задает имя таблицы в базе данных, с которой ассоциируется данный класс.
* @Entity(name = "user\_roles"): Указывает, что класс является сущностью JPA и должен быть маппирован на таблицу user\_roles в базе данных.
* @AllArgsConstructor: Lombok-аннотация для автоматического создания конструктора с аргументами для всех полей.
* @NoArgsConstructor: Lombok-аннотация для автоматического создания конструктора без аргументов.

**Поля класса**

* id: Уникальный идентификатор роли пользователя, автоматически генерируемый с использованием последовательности.
* userAuthority: Перечисление UserAuthority, представляющее право или авторизацию пользователя.
* user: Связь с сущностью User, представляющей пользователя, которому принадлежит данная роль.

**Аннотации на уровне полей**

* @Id: Указывает, что поле id является первичным ключом.
* @GeneratedValue(generator = "user\_role\_id\_seq", strategy = GenerationType.SEQUENCE): Указывает, что значение для поля id будет автоматически генерироваться с использованием последовательности user\_role\_id\_seq.
* @SequenceGenerator(name = "user\_role\_id\_seq", sequenceName = "user\_role\_id\_seq", allocationSize = 1): Определяет генератор последовательности для поля id.
* @Enumerated: Указывает, что поле userAuthority должно быть маппировано как перечисление.
* @JsonIgnore: Указывает, что поле user не должно быть сериализовано в JSON.
* @ManyToOne: Указывает, что существует связь многие-к-одному с сущностью User.
* @JoinColumn(name = "user\_id"): Указывает имя столбца в таблице user\_roles, который хранит внешний ключ для связи с таблицей User.

Класс WishList

Класс WishList представляет собой сущность, которая хранит список желаемых конфигураций компьютеров для пользователя в системе подбора конфигурации. Он связывается с пользователем через отношение один-к-одному и содержит список компьютеров, связанных с этим списком желаемого.

**Импортируемые библиотеки и аннотации**

* com.fasterxml.jackson.annotation.\*: Набор аннотаций для управления сериализацией и десериализацией JSON с использованием Jackson.
* jakarta.persistence.\*: Набор аннотаций JPA для маппинга класса на таблицу базы данных и указания различных аспектов поведения полей класса.
* lombok.\*: Библиотека Lombok для автоматической генерации стандартных методов, таких как геттеры, сеттеры, конструкторы и другие.

**Аннотации на уровне класса**

* @Data: Lombok-аннотация для генерации всех стандартных методов, таких как геттеры, сеттеры, toString(), equals() и hashCode().
* @AllArgsConstructor: Lombok-аннотация для автоматического создания конструктора с аргументами для всех полей.
* @NoArgsConstructor: Lombok-аннотация для автоматического создания конструктора без аргументов.
* @Getter и @Setter: Lombok-аннотации для генерации геттеров и сеттеров для всех полей.
* @Entity(name = "WishList"): Указывает, что класс является сущностью JPA и должен быть маппирован на таблицу WishList в базе данных.
* @Table(name = "WishList"): Задает имя таблицы в базе данных, с которой ассоциируется данный класс.
* @JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id"): Используется для управления циклическими ссылками при сериализации JSON.

**Поля класса**

* id: Уникальный идентификатор списка желаемого, автоматически генерируемый с использованием последовательности.
* user: Связь с сущностью User, представляющей пользователя, которому принадлежит этот список желаемого.
* computers: Список конфигураций компьютеров, связанных с этим списком желаемого.

**Аннотации на уровне полей**

* @Id: Указывает, что поле id является первичным ключом.
* @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "wishlist\_seq\_gen"): Указывает, что значение для поля id будет автоматически генерироваться с использованием последовательности wishlist\_seq.
* @SequenceGenerator(name = "wishlist\_seq\_gen", sequenceName = "wishlist\_seq", allocationSize = 1): Определяет генератор последовательности для поля id.
* @Column(name = "id"): Задает имя столбца в таблице базы данных для поля id.
* @JsonIgnore: Указывает, что поле не должно быть сериализовано в JSON, чтобы избежать рекурсивных ссылок.
* @OneToOne и @JoinColumn(name = "user\_id"): Указывает отношение один-к-одному с таблицей User.
* @OneToMany(mappedBy = "wishList", fetch = FetchType.EAGER): Указывает отношение один-ко-многим с таблицей Computer.