**Документация на интерфейса OrderRepository**

**Общ преглед**

**OrderRepository** е интерфейс предназначен за управление на данните за поръчки в базата данни чрез Spring Data JPA. Той улеснява реализацията на CRUD операции (създаване, четене, обновяване и изтриване) за обекти от тип **Order** без допълнителен код.

**Анотации и импортиране**

* **@Repository**: Специфицира, че този интерфейс е компонент на Spring, който функционира като репозиторий в контекста на Spring Data.
* **import org.springframework.data.jpa.repository.JpaRepository**: Включва функционалността на **JpaRepository**, което позволява лесен достъп до базовите методи за управление на данни.

**Основни методи**

Като наследник на **JpaRepository**, **OrderRepository** автоматично наследява редица методи за работа с данни:

* **save(S entity)**: Записва обект от тип **Order** в базата данни.
* **findById(ID id)**: Търси поръчка по уникално идентификационно число.
* **deleteById(ID id)**: Изтрива поръчка по идентификационно число.
* **findAll()**: Връща списък на всички поръчки в базата данни.

Here's a documentation in Bulgarian for the **OrderItemRepository** interface that you provided:

**Документация на интерфейса OrderItemRepository**

**Общ преглед**

**OrderItemRepository** е интерфейс, който се използва за управление на достъпа до данни за обекти от тип **OrderItem** в базата данни. Този интерфейс наследява **JpaRepository**, което му предоставя набор от методи за CRUD операции (Create, Read, Update, Delete) и позволява лесно да се извършват базови операции с данни без нужда от допълнително кодиране.

**Анотации и импортиране**

* **@Repository**: Анотация, която указва, че интерфейсът е Spring Bean и играе ролята на репозиторий в контекста на Spring Data.
* **import org.springframework.data.jpa.repository.JpaRepository**: Импортиране на **JpaRepository**, което е част от Spring Data JPA и предоставя функционалност за работа с данни в базата.
* **import org.springframework.stereotype.Repository**: Импортиране на анотацията **Repository**.

**Основни методи**

Благодарение на наследяването от **JpaRepository**, **OrderItemRepository** включва редица полезни методи:

* **save(S entity)**: Запазва даден ентити и връща съхранената версия.
* **findById(ID id)**: Връща ентити по зададено ID, ако съществува.
* **deleteById(ID id)**: Изтрива ентити по ID.
* **findAll()**: Връща списък на всички ентити в репозитория.

Документация за интерфейса **ProductRepository**, който представлява репозиториен слой в проект за онлайн магазин, използващ Spring Data JPA за управление на продуктовите данни:

**Общо Описание:**

**ProductRepository** е интерфейс, който наследява **JpaRepository** от Spring Data, осигуряващ готови до използване методи за CRUD (създаване, четене, обновяване и изтриване) операции за обекти от тип **Product**. Това включва автоматично генерирани методи като **findAll()**, **findById()**, **save()**, и **delete()**.

**Основни Методи:**

* **findAll()**: Връща списък от всички продукти в базата данни.
* **findById(Long id)**: Връща продукт по зададен идентификатор.
* **save(Product product)**: Съхранява или обновява продукт в базата данни.
* **delete(Product product)**: Изтрива продукт от базата данни.

**Използване:**

Интерфейсът се използва за директен достъп и манипулация на продуктови данни в базата данни. Той се интегрира лесно с други компоненти на Spring, като сервисни слоеве, които управляват бизнес логиката на приложението.

**ProductRepository** улеснява разработката, като избягва необходимостта от ръчно писане на DAO слой (Data Access Object) и предоставя мощни функционалности за работа с данни, включително пейджиниране и сортиране, които могат да бъдат лесно добавени чрез параметри на методите.

**Персонализирани Заявки:**

Ако са необходими специализирани заявки, които не се покриват от стандартните методи, **ProductRepository** може да се разшири с персонализирани методи, използващи Spring Data JPA анотации като **@Query** за дефиниране на SQL или JPQL заявки.

Този интерфейс е критичен компонент в архитектурата на приложението, който спомага за управлението на продуктови данни в ефективен и скалируем начин, осигурявайки бърз достъп и обработка на данните съгласно бизнес правилата на системата.

Документация за интерфейса **RoleRepository**, който представлява репозиториен слой в проект за онлайн магазин, използващ Spring Data JPA за управление на ролите на потребителите:

**Общо Описание:**

**RoleRepository** е интерфейс, който наследява **JpaRepository** от Spring Data. Този интерфейс осигурява стандартни CRUD операции за обекти от тип **Role** и добавя функционалност за търсене на роля по име.

**Основни Методи:**

* **findAll()**: Връща списък от всички роли в базата данни.
* **findById(Long id)**: Връща роля по зададен идентификатор.
* **save(Role role)**: Съхранява или обновява роля в базата данни.
* **delete(Role role)**: Изтрива роля от базата данни.
* **findByName(String name)**: Търси и връща роля по зададено име.

**Използване:**

Интерфейсът се използва за директен достъп и манипулация на данни за ролите в системата. Той улеснява администрирането на ролите, като позволява бързо добавяне, модификация и изтриване на роли, както и търсене на специфични роли по име за проверка на права и привилегии на потребителите.

**RoleRepository** играе важна роля в управлението на сигурността в приложението, като улеснява присвояването на подходящи роли на потребителите в зависимост от техните задачи и отговорности.

**Персонализирани Заявки:**

Докато **findByName** е пример за персонализирана функционалност, която е добавена към репозитория за удовлетворяване на специфични бизнес изисквания, допълнителни заявки могат да бъдат дефинирани чрез използването на анотацията **@Query**, ако се налага повече специфична логика за достъп до данни.

Този интерфейс е ключов за поддържане на контролиран и сигурен достъп в рамките на системата, гарантирайки, че потребителите имат правилните роли и привилегии за извършване на своите функции в рамките на онлайн магазина.

Документация за интерфейса `UserRepository`, който представлява репозиториен слой в проект за онлайн магазин, използващ Spring Data JPA за управление на потребителските данни:

Документация за интерфейса **UserRepository**, който представлява репозиториен слой в проект за онлайн магазин, използващ Spring Data JPA за управление на потребителските данни:

**Общо Описание:**

**UserRepository** е интерфейс, който наследява **JpaRepository** от Spring Data, предоставяйки стандартни CRUD операции за обекти от тип **User**. Той включва допълнителна функционалност за търсене на потребител по електронен адрес.

**Основни Методи:**

* **findAll()**: Връща списък от всички потребители в базата данни.
* **findById(Long id)**: Връща потребител по зададен идентификатор.
* **save(User user)**: Съхранява или обновява потребител в базата данни.
* **delete(User user)**: Изтрива потребител от базата данни.
* **findByEmail(String email)**: Търси и връща потребител по зададен електронен адрес.

**Използване:**

Интерфейсът **UserRepository** се използва за директен достъп и манипулация на потребителски данни в базата данни. Той е инструментален за функции като регистрация, вход в системата и управление на потребителски профили, като позволява бързо и ефективно управление на потребителската информация.

**UserRepository** улеснява проверката за съществуващи потребителски електронни адреси при регистрация, осигурявайки, че не се допускат дубликати, което е важно за поддържане на уникалност и сигурност на потребителските акаунти.

**Персонализирани Заявки:**

Методът **findByEmail** е пример за персонализирана заявка, която улеснява търсенето на потребители по техния електронен адрес. Ако се налагат други специфични заявки, могат да бъдат добавени допълнителни методи или използвани анотации като **@Query** за дефиниране на специализирани SQL или JPQL заявки в зависимост от нуждите на бизнеса.

Този интерфейс е критичен за управлението на потребителски данни в онлайн магазин, като помага за осигуряване на сигурен и ефективен достъп до информацията, необходима за функционирането на множество потребителски процеси в системата.