Rendeléskezelő Projekt Dokumentáció

1. Bevezetés

A rendeléskezelő projekt célja egy olyan webalkalmazás létrehozása, amely lehetővé teszi éttermek vagy online boltok számára a rendelések hatékony kezelését és nyomon követését. Az alkalmazás a Spring keretrendszert használja a backend fejlesztéséhez és REST API végpontokat biztosít a kommunikációhoz.

2. Felhasználók és Szerepkörök

Az alkalmazásnak két fő felhasználói szerepköre van:

- Vásárló: A vásárlók böngészhetik a termékeket, hozzáadhatják azokat a kosarukhoz, rendeléseket adhatnak le, és nyomon követhetik a rendeléseik státuszát.
- Rendeléskezelő: A rendeléskezelők láthatják az összes aktív rendelést, frissíthetik a rendelések státuszát, és naplózhatják a rendelések teljesítését.

3. Funkcionális Követelmények

3.1 Felhasználókezelés

- Regisztráció és bejelentkezés: A felhasználóknak lehetőségük van regisztrálni és bejelentkezni a rendszerbe. A regisztráció során meg kell adniuk a nevüket, e-mail címüket, és jelszavukat.
- Felhasználói szerepkörök: Az alkalmazásnak támogatnia kell a vásárló és rendeléskezelő szerepköröket. A regisztráció során a felhasználóknak meg kell adniuk, hogy melyik szerepkört választják.

3.2 Vásárlói Funkcionalitás

- Termékek böngészése: A vásárlók böngészhetik a rendelkezésre álló termékeket kategóriákba rendezve és leírással ellátva.
- Kosár kezelése: A vásárlók hozzáadhatnak, módosíthatnak, és törölhetik a termékeket a kosarukból.
- Rendelés feladása: A vásárlók rendelést adhatnak le a kosaruk tartalmából, megadva a kiszállítási címet és egy megjegyzést.
- Rendelések nyomon követése: A vásárlók megtekinthetik és nyomon követhetik a korábbi rendeléseik státuszát.

3.3 Rendeléskezelői Funkcionalitás

- Aktív rendelések megtekintése: A rendeléskezelők láthatják az összes aktív rendelést különböző státuszokkal.

- **Rendelés státusz frissítése:** A rendeléskezelők frissíthetik a rendelések státuszát elfogadva, készítés alatt, kiszállítás alatt stb.
- **Rendelések teljesítése:** A rendeléskezelők naplózhatják a rendelések teljesítését, beleértve a kiszállítás időpontját és egyéb információkat.

3.4 Termékkezelés

- **Termékek hozzáadása:** A rendeléskezelők új termékeket adhatnak hozzá a rendszerhez, megadva a nevet, leírást, árat és egy képet.
- **Termékek szerkesztése és törlése:** A rendeléskezelők szerkeszthetik és törölhetik a meglévő termékeket.

3.5 Rendelési Előzmények

- **Rendelési előzmények megtekintése:** Minden felhasználó hozzáférhet a korábbi rendeléseik részleteihez.

4. REST API Végpontok

Az alkalmazás a következő REST API végpontokat biztosítja:

- **GET /api/products:** Az összes termék lekérdezése.
- **POST /api/products:** Új termék hozzáadása.
- **GET /api/products/{id}:** Egy termék lekérdezése.
- **PUT /api/products/{id}:** Egy termék szerkesztése.
- **DELETE /api/products/{id}:** Egy termék törlése.
- **GET /api/orders:** Az összes rendelés lekérdezése.
- **POST /api/orders:** Új rendelés felvétele.
- **GET /api/orders/{id}:** Egy rendelés lekérdezése.
- **PUT /api/orders/{id}:** Egy rendelés státuszának frissítése.
- **GET /api/cart:** A kosár tartalmának lekérdezése.
- **POST /api/cart:** Termék hozzáadása a kosárhoz.
- **DELETE /api/cart:** Termék eltávolítása a kosárból.
- **GET /api/history:** A rendelési előzmények lek

CartController Dokumentáció

A `CartController` egy Spring Boot alapú RESTful alkalmazás kontrollerje, amely a kosárral kapcsolatos műveleteket végzi. Az alábbiakban a kontroller működését és végpontjait dokumentálom.

1. Végpontok

1.1 GET /api/cart

- Leírás: Visszaadja a felhasználó kosarát vagy egy adott kosarat az azonosító alapján.
- Paraméterek:
  - `userId` (opcionális): A felhasználó azonosítója.
  - `id` (opcionális): A kosár azonosítója.
- Visszatérési érték:
  - Siker esetén visszaadja a kosarát vagy az adott kosarat `CartDto` formájában.
  - Hiba esetén HTTP 400 Bad Request vagy HTTP 500 Internal Server Error.

1.2 POST /api/cart

- Leírás: Termék hozzáadása a kosárhoz vagy új kosár létrehozása.
- Paraméterek:
  - `id` (opcionális): A meglévő kosár azonosítója, amelyhez hozzáadunk terméket.
  - `pId` (opcionális): A termék azonosítója, amelyet hozzáadunk a kosárhoz.
  - `qty` (opcionális): A hozzáadandó termék mennyisége.
  - `userId` (opcionális): Az új kosárhoz tartozó felhasználó azonosítója.
- Visszatérési érték:
  - Siker esetén visszaadja a kosarát `CartDto` formájában.
  - Hiba esetén HTTP 400 Bad Request vagy HTTP 500 Internal Server Error.

1.3 DELETE /api/cart

- Leírás: Termék eltávolítása a kosárból.
- Paraméterek:
  - `cId`: A kosár azonosítója.
  - `cartItemId`: A kosárban található elem azonosítója.
- Visszatérési érték:
  - Siker esetén HTTP 200 OK.
  - Hiba esetén HTTP 500 Internal Server Error.

2. Modell

2.1 CartDto

```
java
public class CartDto {
    // Kosárhoz tartozó adatok
}
```

2.2 CartItemDto

```
java
public class CartItemDto {
```

```
    // Kosár elemeihez tartozó adatok
}
```

3. Szolgáltatások

A `CartController` függ a következő szolgáltatásoktól:

- `CartService`: A kosárhoz kapcsolódó logikát végzi.
- `ProductService`: A termékekkel kapcsolatos logikát végzi.

4. Hibakezelés

A kontroller megfelelő hibakódokkal és hibaüzenetekkel válaszol, amelyek segítik az alkalmazás klienseit a hibák megértésében és kezelésében.

5. Naplózás

A kontroller naplózza az eseményeket, például a sikeres vagy sikertelen kéréseket, és a naplóbejegyzések segítenek az alkalmazás állapotának monitorozásában és hibakeresésében.

6. Biztonság

A végpontokat megfelelő módon védi a Spring Security révén, és csak a megfelelő jogosultságokkal rendelkező felhasználók érhetik el őket.

Ez a dokumentáció áttekintést nyújt a `CartController` működéséről és végpontjairól. A további részletekért és a modell osztályok pontos tartalmáért javasolt a forráskódot és a teszteket megvizsgálni.

OrderController Dokumentáció

A `OrderController` egy Spring Boot alapú RESTful alkalmazás kontrollerje, amely a rendelésekkel kapcsolatos műveleteket végzi. Az alábbiakban a kontroller működését és végpontjait dokumentálom.

1. Végpontok

1.1 GET /api/orders

- Leírás: Visszaadja az összes rendelést, egy adott rendelést azonosító alapján, vagy egy felhasználóhoz tartozó összes rendelést.
- Paraméterek:
  - `id` (opcionális): A rendelés azonosítója.
  - `userId` (opcionális): A felhasználó azonosítója.
- Visszatérési érték:
  - Siker esetén visszaadja a rendeléseket `OrderDto` listaként.
  - Hiba esetén HTTP 400 Bad Request vagy HTTP 500 Internal Server Error.

1.2 PUT /api/orders

- Leírás: Frissíti a rendelés státuszát az azonosító alapján.
- Paraméterek:
  - `id`: A rendelés azonosítója.
  - `statusDto`: Az új rendelési státusz.
- Visszatérési érték:
  - Siker esetén visszaadja a frissített rendelést `OrderDto` formájában.
  - Hiba esetén HTTP 404 Not Found vagy HTTP 500 Internal Server Error.

1.3 POST /api/orders

- Leírás: Létrehoz egy új rendelést a megadott adatok alapján.
- Paraméterek:
  - `orderRequestDto`: A rendelés létrehozásához szükséges adatokat tartalmazza.
- Visszatérési érték:
  - Siker esetén visszaadja az újonnan létrehozott rendelést `OrderDto` formájában.
  - Hiba esetén HTTP 400 Bad Request vagy HTTP 500 Internal Server Error.

2. Modell

2.1 OrderDto

```
java
public class OrderDto {
    // Rendeléshez tartozó adatok
}
```

2.2 OrderStatusDto

```
java
public class OrderStatusDto {
    // Rendelés státuszhoz tartozó adatok
}
```

2.3 `OrderRequestDto`

```
java
public class OrderRequestDto {
    // Rendelés létrehozásához szükséges adatok
}
```

3. Szolgáltatások

A `OrderController` függ a következő szolgáltatásoktól:

- `OrderService`: A rendelésekkel kapcsolatos logikát végzi.

4. Hibakezelés

A kontroller megfelelő hibakódokkal és hibaüzenetekkel válaszol, amelyek segítik az alkalmazás klienseit a hibák megértésében és kezelésében.

5. Naplózás

A kontroller naplózza az eseményeket, például a sikeres vagy sikertelen kéréseket, és a naplóbejegyzések segítenek az alkalmazás állapotának monitorozásában és hibakeresésében.

6. Biztonság

A végpontokat megfelelő módon védi a Spring Security révén, és csak a megfelelő jogosultságokkal rendelkező felhasználók érhetik el őket.

Ez a dokumentáció áttekintést nyújt a `OrderController` működéséről és végpontjairól. A további részletekért és a modell osztályok pontos tartalmáért javasolt a forráskódot és a teszteket megvizsgálni.

ProductController Dokumentáció

A `ProductController` egy Spring Boot alapú RESTful alkalmazás kontrollerje, amely a termékekkel kapcsolatos műveleteket végzi. Az alábbiakban a kontroller működését és végpontjait dokumentálom.

1. Végpontok

1.1 GET /api/products

- Leírás: Visszaadja az összes terméket.
- Visszatérési érték:
  - Siker esetén visszaadja a termékeket `ProductDto` listaként.
  - Hiba esetén HTTP 500 Internal Server Error.

1.2 GET /api/products/{id}

- Leírás: Visszaadja a megadott azonosítójú terméket.
- Paraméterek:
  - `id`: A termék azonosítója.
- Visszatérési érték:
  - Siker esetén visszaadja a terméket `ProductDto` formájában.
  - Hiba esetén HTTP 404 Not Found vagy HTTP 500 Internal Server Error.

1.3 POST /api/products

- Leírás: Létrehoz egy új terméket a megadott adatok alapján.
- Paraméterek:
  - `productDto`: A termék létrehozásához szükséges adatokat tartalmazza.
- Visszatérési érték:
  - Siker esetén visszaadja az újonnan létrehozott terméket `ProductDto` formájában.
  - Hiba esetén HTTP 500 Internal Server Error.

1.4 PUT /api/products/{id}

- Leírás: Frissíti a megadott azonosítójú termék adatait.
- Paraméterek:
  - `id`: A termék azonosítója.
  - `productDto`: A frissítendő termék adatai.
- Visszatérési érték:
  - Siker esetén visszaadja a frissített terméket `ProductDto` formájában.
  - Hiba esetén HTTP 404 Not Found vagy HTTP 500 Internal Server Error.

1.5 DELETE /api/products/{id}

- Leírás: Törli a megadott azonosítójú terméket.
- Paraméterek:
  - `id`: A termék azonosítója.
- Visszatérési érték:
  - Siker esetén HTTP 200 OK.

- Hiba esetén HTTP 500 Internal Server Error.

1.6 `GET /api/products/search`

- Leírás: Keresés a termékek között név alapján.
- Paraméterek:
  - `name`: A keresendő termék neve.
- Visszatérési érték:
  - Siker esetén visszaadja a találatokat `ProductDto` listaként.
  - Hiba esetén HTTP 500 Internal Server Error.

2. Modell

2.1 `ProductDto`

```
java
public class ProductDto {
    // Termékhez tartozó adatok
}
```

3. Szolgáltatások

A `ProductController` függ a következő szolgáltatástól:

- `ProductService`: A termékekkel kapcsolatos logikát végzi.

4. Hibakezelés

A kontroller megfelelő hibakódokkal és hibaüzenetekkel válaszol, amelyek segítik az alkalmazás klienseit a hibák megértésében és kezelésében.

5. Naplózás

A kontroller naplózza az eseményeket, például a sikeres vagy sikertelen kéréseket, és a naplóbejegyzések segítenek az alkalmazás állapotának monitorozásában és hibakeresésében.

6. Biztonság

A végpontokat megfelelő módon védi a Spring Security révén, és csak a megfelelő jogosultságokkal rendelkező felhasználók érhetik el őket.

Ez a dokumentáció áttekintést nyújt a `ProductController` működéséről és végpontjairól. A további részletekért és a modell osztályok pontos tartalmáért javasolt a forráskódot és a teszteket megvizsgálni.