Eindproject – Entreprise Applications

1. Inleiding

Dit document bevat de technische documentatie van EindProject_Adrien, een full-stack webapplicatie gebouwd met Spring Boot. Met deze applicatie kunnen gebruikers producten bekijken, ze toevoegen aan hun winkelmandje, bestellingen plaatsen en een account aanmaken of inloggen.

De applicatie is opgezet volgens de klassieke lagenarchitectuur van Spring Boot. Dat betekent dat controllers, services, repositories en modellen elk hun eigen duidelijke taak hebben. Die scheiding zorgt ervoor dat de code gestructureerd blijft, makkelijk uit te breiden is en goed te onderhouden blijft.

Het doel van deze applicatie is om de kernprincipes te tonen van een moderne, Java-gebaseerde webapplicatie. Denk daarbij aan zaken zoals authenticatie, databasebeheer, de MVC-architectuur en het verwerken van gebruikersinput. In deze documentatie worden de projectstructuur, de belangrijkste onderdelen en de werking van de centrale businesslogica stap voor stap toegelicht.

2. Technologieën en Tools

Het project maakt gebruik van verschillende technologieën en frameworks die samen een robuuste, veilige en gebruiksvriendelijke applicatie vormen. Hieronder volgt een overzicht van de gebruikte stack.

Programmeertaal & Framework

- Java 17+
  De gebruikte programmeertaal voor backendlogica.
- Spring Boot
  Het belangrijkste framework dat instaat voor:
  - Dependency injection
  - MVC-architectuur
  - Routing
  - Security
  - Data persistence
- Spring Web (Spring MVC)
  Handelt webrequests af en koppelt deze aan controllers.
- Spring Security
  Verantwoordelijk voor login, hashing van wachtwoorden en autorisatie
- Spring Data JPA
  Automatische JPA-functionaliteit voor repositories en databasequeries

Frontend en Templating

- Thymeleaf
  Template-engine voor dynamische HTML-pagina's.
- Bootstrap 5
  Voor de styling en lay-out van de gebruikersinterface.

- HTML/CSS
  Basistechnologieën voor de presentatie van de applicatie.

Database en Opslag

- H2
  Voor het opslaan van gebruikers, producten, categorieën, bestellingen en winkelmand-items.
- JPA Entities
  Modelleren van databaseobjecten in Java.

Tools en ontwikkelomgeving

- IntelliJ IDEA
  Voor het ontwikkelen, testen en runnen van de applicatie.
- Maven
  Build tool voor dependencies, packaging en projectstructuur.
- Spring Boot DevTools
  Voor sneller herladen tijdens ontwikkeling.
- Git en Github
  Voor versiebeheer en samenwerking.

Security en hashing

- PasswordEncoder (Bcrypt)
  Wachtwoorden worden nooit in plain-tekst opgeslagen.
  Spring Security zorgt voor hashing en validatie.

Seed-data via DataInitializer

Het project bevat een DataInitializer waarmee standaarddata automatisch in de database wordt geplaatst bij opstart. Dit omvat:

- Een standaardgebruiker
- Categorieën
- Startproducten

Hierdoor is de applicatie meteen bruikbaar zonder de manuele invoer.

3. Architectuur en Laagopbouw

De applicatie volgt een klassieke Spring Boot lagenarchitectuur. Elke laag heeft een duidelijke verantwoordelijkheid:

- Controllelaag
  Verwerkt HTTP-verzoeken en toont de juiste pagina's.
- Servicelaag
  Bevat de businesslogica, zoals winkelmandfunctionaliteit.
- Repositorylaag
  Communiceert met de database via Spring Data JPA.
- Modellaag
  JPA-entiteiten die de tabellen voorstellen.
- Configuratielaag
  Security-instellingen, data-initialisatie en algemene configuratie.

4. Overzicht van de Packages

Model

Bevat alle entiteiten zoals User, Product, Order, Category en CartItem.

Deze klassen vormen de basis van de database-structuur.

Repository

JPA-interfaces zoals UserRepository, ProductRepository, OrderRepository, CartItemRepository.

Ze leveren CRUD-functionaliteit zonder extra code.

Service

Bevat de businesslogica.

Belangrijkste services:

- CartService – beheer winkelmandje
- CustomUserDetailService – gebruikers laden voor login

Controller

Controllers voor alle webpagina's.

Voorbeelden:

- AuthController – registratie en login
- CartController – winkelmandje
- CatalogController - productoverzicht
- HomeController – homepagina

5. Belangrijkste klassen en Complexe Methodes

In dit hoofdstuk worden enkel de klassen en methodes toegelicht die werkelijke logica bevatten. Getters, Setters en eenvoudige mappings worden bewust niet gedocumenteerd.

CartService

Doel: beheer van het winkelmandje voor een gebruiker.

Belangrijkste methodes:

- **addToCart**(User user, Long productId, int quantity, int rentalDays)
  Voeg een product toe aan het winkelmandje. Zoekt het product op, controleert of het al bestaat in de cart en verhoogt anders het aantal.
- **removeItem**(User user, Long cartItemId)
  Verwijdert één specifiek item uit de cart.
- **getCartItems**(User user)
  Haalt alle items op voor een gebruiker.
- **getCartTotal** (User user)
  Berekent het totaalbedrag van alle items in de cart.

CustomUserDetailsService

Doel: koppelt Spring Security aan de User-database

Belangrijkste methode: loadUserByUsername(String username)

Zoekt een gebruiker op via username en geeft een Spring Security UserDetails terug. Nodig voor login en role-management.

AuthController

Doel: afhandeling van login-en registratieschermen.

Belangrijkste methodes:

- **showRegisterForm**(Model model)
  Toont de registratiepapgina
- **registerUser**(User user)
  Maakt een nieuwe gebruiker aan, versleutelt het wachtwoord en slaat hem op.

CartController

Doel: Verwerkt alle acties rond het winkelmandje en de checkout voor de ingelogde gebruiker.

Belangrijkste methodes:

- **showCart**(Model model Authentication authentication)
  Haalt cart-items op voor de ingelogde gebruiker en toont de cart-pagina.
- **addToCart**(...)
  Voegt een product met opgegeven aantal en huurperiode toe aan de cart van de huidige gebruiker. Category -en zoekfilters worden via RedirectAttributes terug meegegeven naar de catalogus

- **removeCartItem**(long cartItemId, Authentication authentication)
  Verwijdert een item uit de cart via de service.
- **updateCartItem**(Long cartItemId, int quantity, Authentication authentication)
  Past de hoeveelheid van een cart-item aan voor de huidige gebruiker.
- **showCheckout**(Model model, Authentication authentication)
  Toont de checkout-pagina met de huidige cart-items en het totaalbedrag. Als de cart leeg is, wordt teruggestuurd naar /cart.
- **confirmCheckout**(Authentication authentication, Model model)
  Controleert of de cart leeg is, maakt een nieuwe Order aan, koppelt alle cart-items aan die order en slaat ze op.
  Toont daarna de confirmation-pagina met order, items en totaalbedrag.

CatalogController

Doel: Toont de productcatalogus, met optionele filters op categorie en zoekterm.

Belangrijkste methodes:

- **showCatalog**(Long categoryId, String search, Model model)
  Haalt alle categorieën op en laadt de juiste lijst van producten, afhankelijk van de filters:

HomeController

Doel: basisnavigatie

Belangrijkste methode: **home**()

Laadt de startpagina van de webshop

6. Broncode

https://github.com/AdrienG0/EindProject.git