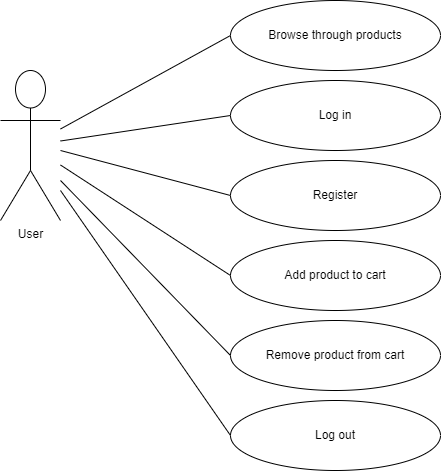
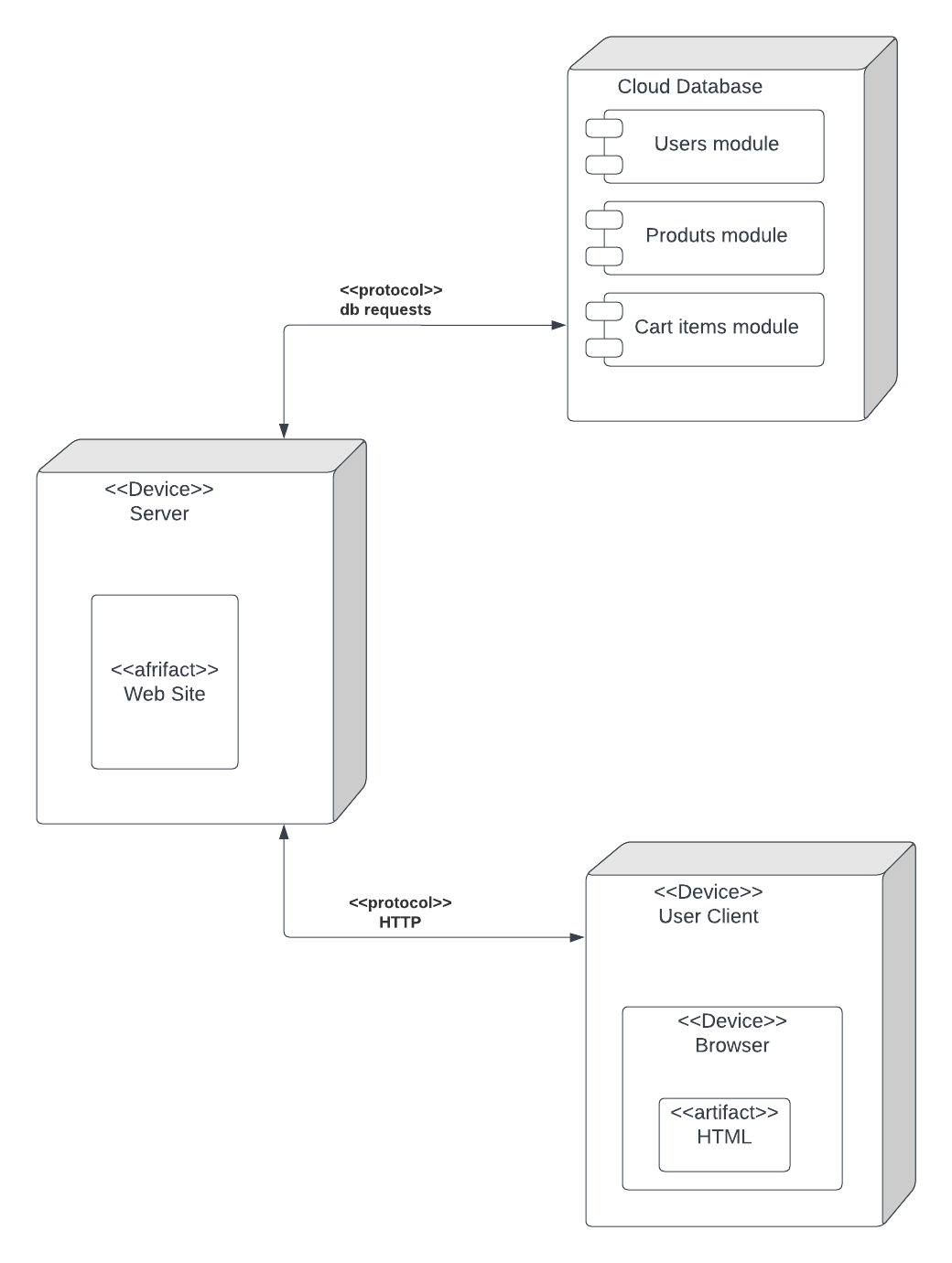
E-Commerce web application

# 1. Design

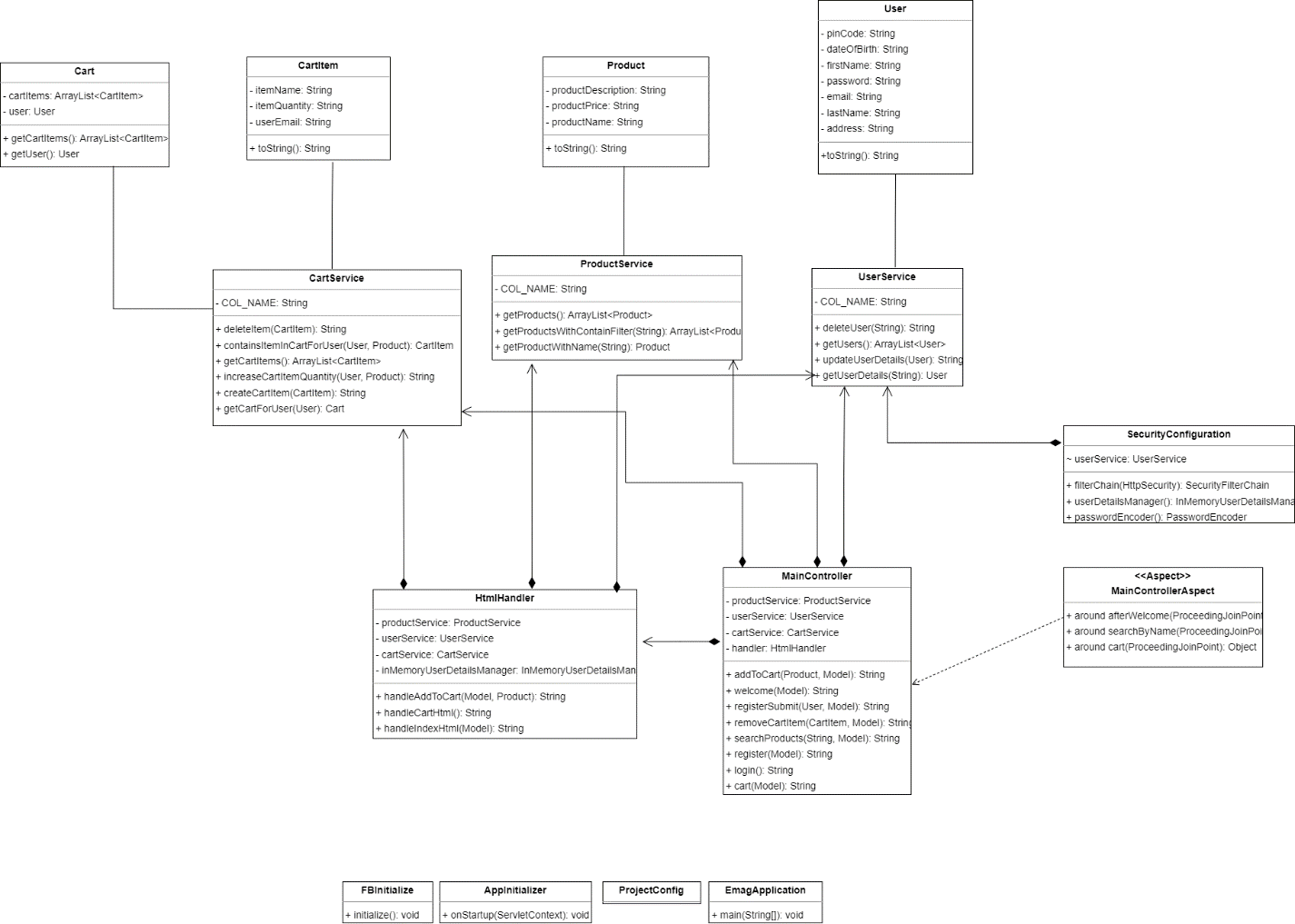
The goal of this project was to create a web application that is focused on the flow a user makes when he uses an online shop (like eMag). The Use Case Diagram is shown below:



The project is based on a client – server architecture on the web. The communication between the two is done via HTTP requests and the server interacts with the database via database requests. The deployment diagram is presented below:



Class diagram:



# 2. Implementation

This application was created using the technologies: Java & Spring for back-end development, HTML, CSS and vanilla JS for front-end development, Thymeleaf for transmitting data from front-end to back-end. From the Spring framework, Spring Security was used in order to create the log in, register and log out functionalities with the help of authorizations. For the sake of transparency and readability, passwords were stored as plain text after the user data was retrieved from the database. Along with Spring Security, Spring AOP was used to inject code before and after certain method calls in order to dynamically show messages tailored for actions that users make (such as searching for a product, logging in, seeing a user’s cart).

The database was implemented using Firebase, which is a service provided by Google that lets users create cloud-based databases, provide authentication within an application, thus removing the responsibility from the developers to the Firebase service, storing information such as photos and many more. The database service that was used in this project is called Firestore, which is a realtime database that stores information grouped up as “collections”, each collection having multiple “documents” and every document has an id and fields. As an example, the project’s database structure can be seen in the following picture:

Graphical user interface, text, application, email

Description automatically generated

The database has 3 collections: “products”, “cart\_items” and “users”, each containing information about the products, all items that are in every cart and every product that is listed. The “products” collection, which is selected in the previous photo, contains several documents (which will be mapped to Product objects once they are retrieved). As shown, every document’s id is auto generated and every product contains the same 3 fields: “productDescription”, “productName” and “productPrice”.

## Source code of the project:

**Aspects package** – contains the MainControllerAspect class, which is responsible with injecting code that runs after some selected methods

package com.emag.aspects;

import org.apache.commons.io.FileUtils;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

import org.jsoup.Jsoup;

import org.jsoup.nodes.Document;

import org.springframework.security.authentication.AnonymousAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.stereotype.Component;

import java.io.File;

import java.io.IOException;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Path;

@Aspect

@Component

public class MainControllerAspect {

@Around("execution(\* com.emag.controller.MainController.welcome(..))")

public Object afterWelcome(ProceedingJoinPoint joinPoint){

Object result = null;

try {

result = joinPoint.proceed(joinPoint.getArgs());

} catch (Throwable e) {

throw new RuntimeException(e);

}

try {

String newText = new String(Files.readAllBytes(Path.of("src/main/resources/templates/newIndex.html")),StandardCharsets.UTF\_8);

Document newDocument = Jsoup.parse(newText);

Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

if(!(authentication instanceof AnonymousAuthenticationToken)) {

String currentUserName = authentication.getName();

newDocument.select("nav").get(0)

.select("div.container-fluid").get(0)

.select("p").get(0).text("Hello, " + currentUserName);

}

else {

newDocument.select("nav").get(0)

.select("div.container-fluid").get(0)

.select("p").get(0).text("Hello, someone");

}

File indexFile = new File("src/main/resources/templates/newestIndex.html");

FileUtils.writeStringToFile(indexFile, newDocument.toString(), StandardCharsets.UTF\_8);

return "newestIndex";

} catch (IOException e) {

return result;

}

}

@Around("execution(\* com.emag.controller.MainController.searchProducts(..))")

public Object searchByName(ProceedingJoinPoint joinPoint) {

String name = joinPoint.getArgs()[0].toString();

System.out.println(name);

Object result = null;

try {

result = joinPoint.proceed(joinPoint.getArgs());

} catch (Throwable e) {

throw new RuntimeException(e);

}

try {

String newText = new String(Files.readAllBytes(Path.of("src/main/resources/templates/newIndex.html")),StandardCharsets.UTF\_8);

Document newDocument = Jsoup.parse(newText);

newDocument.select("nav").get(0)

.select("div.container-fluid").get(0)

.select("p").get(0).text("Search results that contain: " + name);

File indexFile = new File("src/main/resources/templates/newestIndex.html");

FileUtils.writeStringToFile(indexFile, newDocument.toString(), StandardCharsets.UTF\_8);

return "newestIndex";

} catch (IOException e) {

return result;

}

}

@Around("execution(\* com.emag.controller.MainController.cart(..))")

public Object cart(ProceedingJoinPoint joinPoint) {

Object result = null;

try {

result = joinPoint.proceed(joinPoint.getArgs());

} catch (Throwable e) {

throw new RuntimeException(e);

}

try {

String newText = new String(Files.readAllBytes(Path.of("src/main/resources/templates/newCart.html")),StandardCharsets.UTF\_8);

Document newDocument = Jsoup.parse(newText);

Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

if(!(authentication instanceof AnonymousAuthenticationToken)) {

String currentUserName = authentication.getName();

newDocument.select("nav").get(0)

.select("div.container-fluid").get(0)

.select("p").get(0).text(currentUserName + "'s cart");

}

File indexFile = new File("src/main/resources/templates/newestCart.html");

FileUtils.writeStringToFile(indexFile, newDocument.toString(), StandardCharsets.UTF\_8);

return "newestCart";

} catch (IOException e) {

return result;

}

}

}

**Config package** – contains classes that are responsible with the configuration of Spring

AppInitializer class:

package com.emag.config;

import jakarta.servlet.ServletContext;

import org.springframework.web.WebApplicationInitializer;

import org.springframework.web.context.ContextLoaderListener;

import org.springframework.web.context.support.AnnotationConfigWebApplicationContext;

import org.springframework.web.filter.DelegatingFilterProxy;

public class AppInitializer implements WebApplicationInitializer {

@Override

public void onStartup(ServletContext servletContext) {

AnnotationConfigWebApplicationContext root = new AnnotationConfigWebApplicationContext();

root.register(SecurityConfiguration.class);

servletContext.addListener(new ContextLoaderListener(root));

servletContext.addFilter("securityFilter", new DelegatingFilterProxy("springSecurityFilterChain"))

.addMappingForUrlPatterns(null, false, "/");

}

}

ProjectConfig class:

package com.emag.config;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.EnableAspectJAutoProxy;

@Configuration

@ComponentScan(basePackages = {"com.emag.services", "com.emag.aspects", "com.emag.controller"})

@EnableAspectJAutoProxy

public class ProjectConfig {

}

SecurityConfiguration class:

package com.emag.config;

import com.emag.services.UserService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.crypto.password.NoOpPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

public class SecurityConfiguration{

@Autowired

private UserService userService;

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf()

.disable()

.authorizeHttpRequests()

.requestMatchers("/cart")

.authenticated()

.anyRequest()

.permitAll()

.and()

.formLogin()

.and()

.logout().logoutUrl("/logout").logoutSuccessUrl("/").deleteCookies("auth\_code", "JSESSIONID").invalidateHttpSession(true);

return http.build();

}

@Bean

public InMemoryUserDetailsManager userDetailsManager() {

return new InMemoryUserDetailsManager();

}

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance();

}

}

**Controller package** – responsible with processing http methods sent from and to the client side of the application

MainController class:

package com.emag.config;

import com.emag.services.UserService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.crypto.password.NoOpPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

public class SecurityConfiguration{

@Autowired

private UserService userService;

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf()

.disable()

.authorizeHttpRequests()

.requestMatchers("/cart")

.authenticated()

.anyRequest()

.permitAll()

.and()

.formLogin()

.and()

.logout().logoutUrl("/logout").logoutSuccessUrl("/").deleteCookies("auth\_code", "JSESSIONID").invalidateHttpSession(true);

return http.build();

}

@Bean

public InMemoryUserDetailsManager userDetailsManager() {

return new InMemoryUserDetailsManager();

}

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance();

}

}

**Logic package –** contains the HtmlHanlder class, which is responsible with parsing html templates and populating it with objects taken from the database

package com.emag.logic;

import com.emag.models.Cart;

import com.emag.models.CartItem;

import com.emag.models.Product;

import com.emag.models.User;

import com.emag.services.CartService;

import com.emag.services.ProductService;

import com.emag.services.UserService;

import org.apache.commons.io.FileUtils;

import org.jsoup.Jsoup;

import org.jsoup.nodes.Document;

import org.jsoup.nodes.Element;

import org.jsoup.select.Elements;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.authentication.AnonymousAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.crypto.password.NoOpPasswordEncoder;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.stereotype.Service;

import org.springframework.ui.Model;

import java.io.File;

import java.io.IOException;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Path;

import java.util.ArrayList;

import java.util.concurrent.ExecutionException;

@Service

public class HtmlHandler {

@Autowired

private UserService userService;

@Autowired

private CartService cartService;

@Autowired

private ProductService productService;

@Autowired

private InMemoryUserDetailsManager inMemoryUserDetailsManager;

public String handleCartHtml() {

try {

String text = new String(Files.readAllBytes(Path.of("src/main/resources/templates/cart.html")), StandardCharsets.UTF\_8);

Document document = Jsoup.parse(text);

Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

if(!(authentication instanceof AnonymousAuthenticationToken)) {

String currentUserName = authentication.getName();

document.select("button#loginButton").get(0).remove();

document.select("button#registerButton").get(0).remove();

double totalPrice = 0;

User currentUser = userService.getUserDetails(authentication.getName());

Cart cart = cartService.getCartForUser(currentUser);

Elements elements = document.select("div.card-body").get(0).select("div.row");

elements.get(0).remove();

for (CartItem cartItem: cart.getCartItems()) {

Element cartItemDocument = elements.get(0).clone();

Product product = productService.getProductWithName(cartItem.getItemName());

Elements dataParagraphs = cartItemDocument.select("div#data").get(0).select("input");

dataParagraphs.get(0).attr("placeholder",cartItem.getUserEmail());

dataParagraphs.get(1).attr("placeholder",cartItem.getItemName());

cartItemDocument.select("div#data").select("p").get(1).text("Description: " + product.getProductDescription());

cartItemDocument.select("div#data").get(0).removeAttr("id");

Elements nrParagraphs = cartItemDocument.select("div#nr").get(0).select("input");

nrParagraphs.get(0).attr("placeholder",cartItem.getItemQuantity());

double price = Integer.parseInt(cartItem.getItemQuantity()) \* Double.parseDouble(product.getProductPrice());

totalPrice += price;

cartItemDocument.select("div#nr").get(0).select("p").get(1).text("Price: $" + price);

cartItemDocument.select("div#nr").get(0).removeAttr("id");

document.select("div.card-body").get(0).append(cartItemDocument.toString());

document.select("div.card-body").get(0).append("<hr class=\"my-4\" />");

}

Elements prices = document.select("div.col-md-4").get(0).select("div.card").get(0).select("div.card-body").get(0).select("ul").get(0).select("li");

prices.get(0).select("span").get(0).text("$"+totalPrice);

prices.get(2).select("span").get(0).text("$"+totalPrice);

if(cart.getCartItems().size()==0){

Element ul = document.select("ul.list-group").get(0);

ul.select("li").get(0).remove();

ul.select("li").get(0).remove();

ul.select("li").get(0).select("div").get(0).select("strong").get(0).text("The shopping cart");

ul.select("li").get(0).select("div").get(0).select("strong").get(1).select("p").get(0).text("is empty");

ul.select("li").get(0).select("span").get(0).remove();

document.select("button#checkoutButton").get(0).attr("style","display: none;");

}

}

File indexFile = new File("src/main/resources/templates/newCart.html");

FileUtils.writeStringToFile(indexFile, document.toString(), StandardCharsets.UTF\_8);

return "newCart";

} catch (IOException | ExecutionException | InterruptedException e) {

System.out.println("Problem while displaying products");

return "cart";

}

}

public String handleSearchHtml(String name) {

try {

String text = new String(Files.readAllBytes(Path.of("src/main/resources/templates/index.html")),StandardCharsets.UTF\_8);

Document document = Jsoup.parse(text);

Elements elements = document.select("div.card");

elements.get(0).remove();

ArrayList<Product> products = productService.getProductsWithContainFilter(name);

for (Product product: products) {

Element productDocument = elements.get(0).clone();

productDocument.select("div.card-body").get(0).select("input").get(0).attr("placeholder",product.getProductName());

productDocument.select("div.card-body").get(0).select("input").get(0).attr("value",product.getProductName());

productDocument.select("div.card-body").get(0).select("input").get(1).attr("placeholder",product.getProductDescription());

productDocument.select("div.card-body").get(0).select("input").get(1).attr("value",product.getProductDescription());

productDocument.select("ul.list-group").get(0).select("li").get(0).select("input").get(0).attr("placeholder", product.getProductPrice());

productDocument.select("ul.list-group").get(0).select("li").get(0).select("input").get(0).attr("value", product.getProductPrice());

productDocument.addClass("d-inline-block");

productDocument.attr("style","width: 18rem;");

document.select("body").append(productDocument.toString());

}

Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

if(!(authentication instanceof AnonymousAuthenticationToken)) {

String currentUserName = authentication.getName();

System.out.println(currentUserName);

document.select("button#loginButton").get(0).remove();

document.select("button#registerButton").get(0).remove();

}

else {

document.select("button#logoutButton").get(0).remove();

}

File indexFile = new File("src/main/resources/templates/newIndex.html");

FileUtils.writeStringToFile(indexFile, document.toString(), StandardCharsets.UTF\_8);

String newText = new String(Files.readAllBytes(Path.of("src/main/resources/templates/newIndex.html")),StandardCharsets.UTF\_8);

Document newDocument = Jsoup.parse(newText);

return "newIndex";

} catch (IOException e) {

System.out.println("Problem while displaying products");

return "index";

}

}

public String handleAddToCart(Model model, Product product) {

Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

if(!(authentication instanceof AnonymousAuthenticationToken)) {

String currentUserName = authentication.getName();

try {

User currentUser = userService.getUserDetails(currentUserName);

if(cartService.containsItemInCartForUser(currentUser,product)!=null){

cartService.increaseCartItemQuantity(currentUser,product);

}

else {

cartService.createCartItem(new CartItem(currentUserName, product.getProductName(), "1"));

}

} catch (ExecutionException | InterruptedException e) {

throw new RuntimeException(e);

}

}

return handleIndexHtml(model);

}

public String handleIndexHtml(Model model) {

ArrayList<User> users = userService.getUsers();

ArrayList<UserDetails> userDetails = new ArrayList<>();

for (User user: users){

UserDetails userDetailsAux = org.springframework.security.core.userdetails.User.withUsername(user.getEmail())

.password(NoOpPasswordEncoder.getInstance().encode(user.getPassword()))

.roles("USER").build();

userDetails.add(userDetailsAux);

if (!inMemoryUserDetailsManager.userExists(user.getEmail())) {

inMemoryUserDetailsManager.createUser(userDetailsAux);

}

}

try {

String text = new String(Files.readAllBytes(Path.of("src/main/resources/templates/index.html")),StandardCharsets.UTF\_8);

Document document = Jsoup.parse(text);

Elements elements = document.select("div.card");

elements.get(0).remove();

ArrayList<Product> products = productService.getProducts();

for (Product product: products) {

Element productDocument = elements.get(0).clone();

productDocument.select("div.card-body").get(0).select("input").get(0).attr("placeholder",product.getProductName());

productDocument.select("div.card-body").get(0).select("input").get(0).attr("value",product.getProductName());

productDocument.select("div.card-body").get(0).select("input").get(1).attr("placeholder",product.getProductDescription());

productDocument.select("div.card-body").get(0).select("input").get(1).attr("value",product.getProductDescription());

productDocument.select("ul.list-group").get(0).select("li").get(0).select("input").get(0).attr("placeholder", product.getProductPrice());

productDocument.select("ul.list-group").get(0).select("li").get(0).select("input").get(0).attr("value", product.getProductPrice());

productDocument.addClass("d-inline-block");

productDocument.attr("style","width: 18rem;");

document.select("body").append(productDocument.toString());

}

Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

if(!(authentication instanceof AnonymousAuthenticationToken)) {

document.select("button#loginButton").get(0).remove();

document.select("button#registerButton").get(0).remove();

}

else {

document.select("button#logoutButton").get(0).remove();

}

File indexFile = new File("src/main/resources/templates/newIndex.html");

FileUtils.writeStringToFile(indexFile, document.toString(), StandardCharsets.UTF\_8);

String newText = new String(Files.readAllBytes(Path.of("src/main/resources/templates/newIndex.html")),StandardCharsets.UTF\_8);

Document newDocument = Jsoup.parse(newText);

return "newIndex";

} catch (IOException e) {

System.out.println("Problem while displaying products");

return "index";

}

}

}

**Models package** – contains the classes that contain information mapped from the client side of the application or the database.

Cart class:

package com.emag.models;

import java.util.ArrayList;

public class Cart {

private User user;

private ArrayList<CartItem> cartItems;

public Cart(User user, ArrayList<CartItem> cartItems) {

this.user = user;

this.cartItems = cartItems;

}

public User getUser() {

return user;

}

public ArrayList<CartItem> getCartItems() {

return cartItems;

}

}

CartItem class:

package com.emag.models;

public class CartItem {

private String userEmail;

private String itemName;

private String itemQuantity;

public CartItem() {

}

public CartItem(String userEmail, String itemName, String itemQuantity) {

this.userEmail = userEmail;

this.itemName = itemName;

this.itemQuantity = itemQuantity;

}

public String getUserEmail() {

return userEmail;

}

public void setUserEmail(String userEmail) {

this.userEmail = userEmail;

}

public String getItemName() {

return itemName;

}

public void setItemName(String itemName) {

this.itemName = itemName;

}

public String getItemQuantity() {

return itemQuantity;

}

public void setItemQuantity(String itemQuantity) {

this.itemQuantity = itemQuantity;

}

@Override

public String toString() {

return this.userEmail + "item name: " + this.itemName + ", quantity: " + this.itemQuantity;

}

}

Product class:

package com.emag.models;

public class Product {

private String productName;

private String productDescription;

private String productPrice;

public Product() {

}

public Product(String productName, String productDescription, String productPrice) {

this.productName = productName;

this.productDescription = productDescription;

this.productPrice = productPrice;

}

public String getProductName() {

return productName;

}

public void setProductName(String productName) {

this.productName = productName;

}

public String getProductDescription() {

return productDescription;

}

public void setProductDescription(String productDescription) {

this.productDescription = productDescription;

}

public String getProductPrice() {

return productPrice;

}

public void setProductPrice(String productPrice) {

this.productPrice = productPrice;

}

@Override

public String toString() {

return "Product: " + this.productName + ", description: " + this.productDescription + ", price: " + this.productPrice;

}

}

User class:

package com.emag.models;

public class User {

private String firstName;

private String lastName;

private String email;

private String address;

private String dateOfBirth;

private String pinCode;

private String password;

public User() {

}

public User(String firstName, String lastName, String email, String address, String dateOfBirth, String pinCode, String password) {

this.firstName = firstName;

this.lastName = lastName;

this.email = email;

this.address = address;

this.dateOfBirth = dateOfBirth;

this.pinCode = pinCode;

this.password = password;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getDateOfBirth() {

return dateOfBirth;

}

public void setDateOfBirth(String dateOfBirth) {

this.dateOfBirth = dateOfBirth;

}

public String getPinCode() {

return pinCode;

}

public void setPinCode(String pinCode) {

this.pinCode = pinCode;

}

@Override

public String toString() {

return "User: " + this.email + ", " + this.firstName + ", " + this.lastName + ", " + this.address + ", " +

this.dateOfBirth + ", " + this.pinCode + ", " + this.password;

}

}

**Services package –** contains the classes that are responsible with interacting with the database

CartService class:

package com.emag.services;

import com.emag.models.Cart;

import com.emag.models.CartItem;

import com.emag.models.Product;

import com.emag.models.User;

import com.google.api.core.ApiFuture;

import com.google.cloud.firestore.\*;

import com.google.firebase.cloud.FirestoreClient;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.ExecutionException;

@Service

public class CartService {

private static final String COL\_NAME = "cart\_items";

public String createCartItem(CartItem cartItem) throws ExecutionException, InterruptedException {

Firestore dbFirestore = FirestoreClient.getFirestore();

ApiFuture<WriteResult> collectionsApiFuture = dbFirestore.collection(COL\_NAME).document(cartItem.getUserEmail() + cartItem.getItemName()).set(cartItem);

return "Cart item " + cartItem + " added successfully at time: " + collectionsApiFuture.get().getUpdateTime();

}

public ArrayList<CartItem> getCartItems() {

Firestore dbFirestore = FirestoreClient.getFirestore();

ApiFuture<QuerySnapshot> future = dbFirestore.collection(COL\_NAME).get();

try {

List<QueryDocumentSnapshot> documents = future.get().getDocuments();

ArrayList<CartItem> cartItems = new ArrayList<>();

for (QueryDocumentSnapshot document : documents) {

CartItem cartItem = document.toObject(CartItem.class);

cartItems.add(cartItem);

}

return cartItems;

} catch (InterruptedException | ExecutionException e) {

return new ArrayList<>();

}

}

public Cart getCartForUser(User user) {

ArrayList<CartItem> cartItems = getCartItems();

cartItems.removeIf(cartItem -> !cartItem.getUserEmail().equals(user.getEmail()));

return new Cart(user,cartItems);

}

public String increaseCartItemQuantity(User user, Product product) throws ExecutionException, InterruptedException {

Firestore dbFirestore = FirestoreClient.getFirestore();

CartItem item = containsItemInCartForUser(user,product);

int quantity = Integer.parseInt(item.getItemQuantity()) +1;

item.setItemQuantity(Integer.toString(quantity));

ApiFuture<WriteResult> collectionsApiFuture = dbFirestore.collection(COL\_NAME).document(user.getEmail() + item.getItemName())

.set(item);

return "Cart item " + item + " updated successfully at time: " + collectionsApiFuture.get().getUpdateTime();

}

public CartItem containsItemInCartForUser(User user, Product product) {

ArrayList<CartItem> cartItems = getCartForUser(user).getCartItems();

for(CartItem cartItem: cartItems) {

if (cartItem.getItemName().equals(product.getProductName()))return cartItem;

}

return null;

}

public String deleteItem(CartItem cartItem) {

Firestore dbFirestore = FirestoreClient.getFirestore();

dbFirestore.collection(COL\_NAME).document(cartItem.getUserEmail() + cartItem.getItemName()).delete();

return "Document with email " + cartItem.getUserEmail() + cartItem.getItemName() + " has been deleted.";

}

}

FBInitialize class:

package com.emag.services;

import com.google.auth.oauth2.GoogleCredentials;

import com.google.firebase.FirebaseApp;

import com.google.firebase.FirebaseOptions;

import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

@Service

public class FBInitialize {

@PostConstruct

public void initialize() {

try {

FileInputStream serviceAccount =

new FileInputStream("src/main/resources/static/service-account.json");

FirebaseOptions options = new FirebaseOptions.Builder()

.setCredentials(GoogleCredentials.fromStream(serviceAccount))

.setDatabaseUrl("https://ebuy-78b02-default-rtdb.firebaseio.com")

.build();

FirebaseApp.initializeApp(options);

} catch (FileNotFoundException e) {

System.out.println("Service account file not found");

} catch (IOException e) {

throw new RuntimeException(e);

}

}

}

ProductService class:

package com.emag.services;

import com.emag.models.Product;

import com.google.api.core.ApiFuture;

import com.google.cloud.firestore.Firestore;

import com.google.cloud.firestore.QueryDocumentSnapshot;

import com.google.cloud.firestore.QuerySnapshot;

import com.google.firebase.cloud.FirestoreClient;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.ExecutionException;

@Service

public class ProductService {

private static final String COL\_NAME = "products";

public ArrayList<Product> getProducts() {

Firestore dbFirestore = FirestoreClient.getFirestore();

ApiFuture<QuerySnapshot> future = dbFirestore.collection(COL\_NAME).get();

try {

List<QueryDocumentSnapshot> documents = future.get().getDocuments();

ArrayList<Product> products = new ArrayList<>();

for (QueryDocumentSnapshot document : documents) {

Product product = document.toObject(Product.class);

products.add(product);

}

return products;

} catch (InterruptedException | ExecutionException e) {

return new ArrayList<>();

}

}

public ArrayList<Product> getProductsWithContainFilter(String filter){

ArrayList<Product> products = getProducts();

products.removeIf(product -> !product.getProductName().contains(filter));

return products;

}

public Product getProductWithName(String name) {

ArrayList<Product> products = getProducts();

for(Product product: products) {

if (product.getProductName().equals(name))return product;

}

return null;

}

}

UserService class:

package com.emag.services;

import com.emag.models.User;

import com.google.api.core.ApiFuture;

import com.google.cloud.firestore.\*;

import com.google.firebase.cloud.FirestoreClient;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.ExecutionException;

@Service

public class UserService {

private static final String COL\_NAME = "users";

public String saveUserDetails(User user) throws ExecutionException, InterruptedException {

Firestore dbFirestore = FirestoreClient.getFirestore();

ApiFuture<WriteResult> collectionsApiFuture = dbFirestore.collection(COL\_NAME).document(user.getEmail())

.set(user);

return collectionsApiFuture.get().getUpdateTime().toString();

}

public User getUserDetails(String email) throws ExecutionException, InterruptedException {

Firestore dbFirestore = FirestoreClient.getFirestore();

DocumentReference documentReference = dbFirestore.collection(COL\_NAME).document(email);

ApiFuture<DocumentSnapshot> future = documentReference.get();

DocumentSnapshot document = future.get();

User user = null;

if (document.exists()) {

user = document.toObject(User.class);

}

return user;

}

public String updateUserDetails(User user) throws ExecutionException, InterruptedException {

Firestore dbFirestore = FirestoreClient.getFirestore();

ApiFuture<WriteResult> collectionsApiFuture = dbFirestore.collection(COL\_NAME).document(user.getEmail())

.set(user);

return collectionsApiFuture.get().getUpdateTime().toString();

}

public String deleteUser(String email) {

Firestore dbFirestore = FirestoreClient.getFirestore();

dbFirestore.collection(COL\_NAME).document(email).delete();

return "Document with email " + email + " has been deleted.";

}

public ArrayList<User> getUsers() {

Firestore dbFirestore = FirestoreClient.getFirestore();

ApiFuture<QuerySnapshot> future = dbFirestore.collection("users").get();

try {

List<QueryDocumentSnapshot> documents = future.get().getDocuments();

ArrayList<User> users = new ArrayList<>();

for (QueryDocumentSnapshot document : documents) {

User user = document.toObject(User.class);

users.add(user);

}

return users;

} catch (ExecutionException | InterruptedException e) {

return new ArrayList<>();

}

}

}

**EmagApplication** – The class responsible with running the whole application

package com.emag;

]

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class EmagApplication {

public static void main(String[] args) {

SpringApplication.run(EmagApplication.class, args);

}

}