

Cooking Manager

Vývoj aplikácií s viacvrstvovou architektúrou

Ondrej Lesák

Alex Macala

Dávid Kováč

Tomáš Matejov

Adrián Hládek

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

OBSAH

1. Rozdelenie tímu	3
2. Zámer	3
3. Vízia	4
4. Model 5W	4
5. RACI matica	5
6. Používatelia	5
7. Use Cases	6
8. Business model	8
9. Class Diagram	9
10. Package Diagram	9
11. Databázový model	10
12. Funkčné a nefunkčné požiadavky	11
13. Základný návrh UX	12
14. Načítanie pomocou XML	15
15. Funkčnosť aplikácie	17
16. Dátová vrstva	20
17. Login Controller	20
18. Home Controller	21
19. Recipe Controller	24
20. Add Recipe Controller	25
21. UpdateRecipe Controller	27
22. Register.fxml	33
23. Home.fxml	34
24. AddRecipe.fxml	34
25. Záver	35

1. Rozdelenie tímu

Meno a priezvisko	Pozícia
Adrián Hládek	SW Tester
Dávid Kovál'	SW Architect
Ondrej Lesák	Team Leader
Alex Macala	FE developer
Tomáš Matejov	BE developer

2. Zámer

Účelom tohto projektu je vyvinúť desktopovú aplikáciu v jazyku Java, ktorá umožňuje používateľom zadávať, organizovať a vyhľadávať recepty.

Aplikácia by mala používateľom poskytnúť možnosť pridávať a upravovať recepty, ako aj označovať ich podľa ingrediencií, typu jedla alebo diétnych obmedzení. Aplikácia by mala obsahovať aj funkcie na plánovanie jedál a generovanie nákupných zoznamov na základe užívateľom zvolených receptov. Okrem toho aplikácia bude používateľom poskytovať nutričné informácie, ktoré vedia využiť najmä pri vytváraní "fit" jedálneho lístka.

Cieľom je poskytnúť používateľom prívetivý a efektívny spôsob, ako spravovať zbierku receptov a plánovať jedlá. Aplikácia bude navrhnutá tak, aby bola intuitívna a prístupná pre používateľov všetkých úrovní zručností.

Vývojový tím bude využívať Javu na vývoj backendu (Java 17) a frontendu (JavaFX), ako aj vhodný škálovateľný databázový systém (PostgreSQL 14) na ukladanie používateľských údajov. Finálny produkt bude pred vydaním dôkladne otestovaný vo forme expert testingu a následne používateľských testov, aby sa zaistila funkčnosť a vhodná - intuitívna použiteľnosť.

3. Vízia

Víziou nášho projektu je, aby si ľudia medzi sebou vymieňali svoje recepty, podelili sa s nimi s inými ľuďmi. Ďalšou víziou môže takisto byť, že ľudia, ktorí doteraz nevarili, tak sa to práve prostredníctvom našej aplikácie môžu naučiť.

4. Model 5W

a. Kto?

Aplikácia *Cooking manager* je určená pre všetkých ľudí, ktorí sa chcú podeliť o svoje recepty, vyhľadávať recepty. Môže sa jednať o kuchárov, študentov, ale aj o obyčajných ľudí, ktorí majú radi varenie.

Aplikácia bude prístupná a intuitívna pre používateľov všetkých úrovní technických zručností, čo znamená, že sa jednoducho bude vedieť v nej orientovať aj človek s minimálnymi technickými znalosťami.

b. Čo?

Aplikácia *Cooking manager* umožňuje používateľom zadávať, organizovať a vyhľadávať recepty. Aplikácia ponúka možnosť pridávať a upravovať recepty, ako aj filtrovať recepty podľa ingrediencií alebo typu jedla.

Cieľom aplikácie a projektu je poskytnúť používateľom prívetivý a efektívny spôsob, ako spravovať zbierku receptov a plánovať jedlá.

c. Kde?

Aplikácia sa bude môcť používať kdekoľvek, kde je zabezpečený prístup k internetu. Aplikácia bude nastavená primárne v anglickom jazyku a bude nastavená na dostupnosť v krajinách celého sveta.

d. Kedy?

e. Prečo?

Na túto otázku by vedel z časti odpovedať každý používateľ, ktorý použil našu aplikáciu. Pre niekoho môže byť naša aplikácia užitočná pre začiatky varenia a učenia sa ľahkých receptov, pre niekoho to môže byť na získanie správnosti ingrediencií, výmeny receptov, skúsenie nových receptov, porovnanie postupu pri varení určitého receptu, vyhľadanie receptu na základe ingrediencie.

5. RACI matica

Osoba	Dávid Kovál (SW Architect)	Ondrej Lesák (Team leader)	Adrián Hládek (SW Tester)	Alex Macala (FE Developer)	Tomáš Matejov (BE Developer)
Balík prác					
Phase 1 - Research					
Business model	R	A		I	C
Use Case Model	R	A		C	C
Package model	R	A		C	C
Database model	R	RA	I	I	C
Phase 2 - Define					
Design screens	RC	A	I	R	C
Database initialization	I	RA		I	R
Define API endpoints	C	RA	I	C	R
Define Core functionality	C	RA	I	I	R
Phase 3 - Develop					
Create API endpoints	C	RA	I	I	R
Create API controllers	C	RA	I	I	R
Create logger	C	RA	I	I	R
Create views	RC	CA	I	R	I
Expert testing	CA	I	R	I	I
Testing report	CA	I	R	I	I

Druhy zodpovedností:

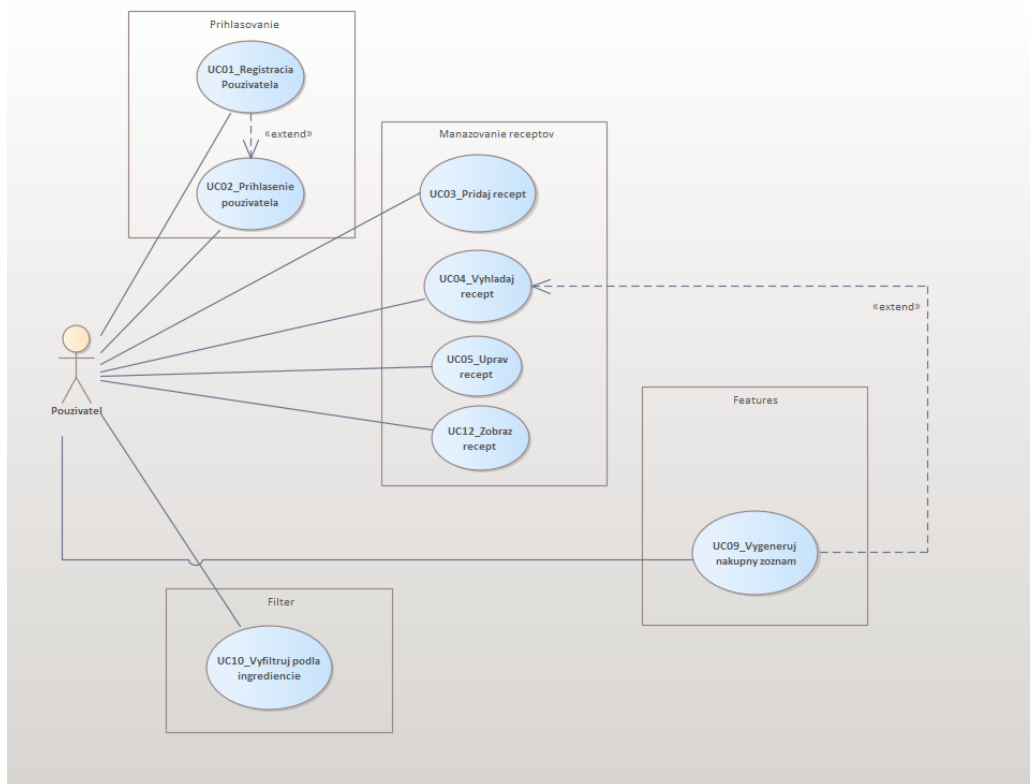
A – Approved/Schvalovanie; R – Realizácia; C – Konzultácia; I - Informovanie

6. Používatelia

V našej aplikácii budeme deliť používateľov na nasledovné typy:

- Admin/šéfkuchár - bude môcť pridávať, editovať a vymazávať všetky recepty
- Kuchár - bude môcť pridávať, editovať a vymazávať svoje recepty a vidieť všetky recepty
- Bežný používateľ - bude môcť vidieť všetky recepty

7. Use Cases

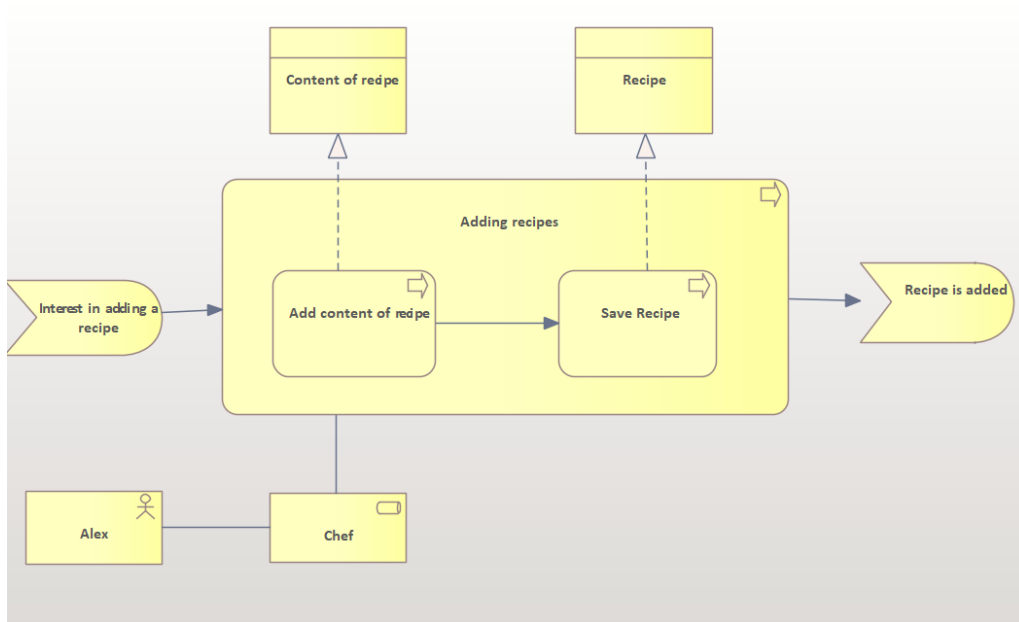


Aplikácia pozostáva z desiatich Use Case-och:

- UC01 Registrácia používateľa
 - 1. Používateľ klikne na tlačidlo Register
 - 2. Používateľ zadá používateľské meno
 - 3. Používateľ zadá emailovú adresu
 - 4. Používateľ si nastaví heslo
 - 5. Používateľ klikne na tlačidlo Registrovať sa
 - 6. Systém uloží dáta do databázy
- UC02 Prihlásenie používateľa
 - 1. Používateľ klikne na tlačidlo Log In
 - 2. Používateľ zadá používateľské meno
 - 3. Používateľ zadá heslo
 - 4. Používateľ klikne na tlačidlo Prihlásiť Sa
- UC03 Pridaj recept
 - 1. Pridanie receptu pomocou XML súboru
 - 1. Používateľ klikne na tlačidlo Vytvoriť nový recept
 - 2. Používateľ klikne na tlačidlo Zvoliť súbor XML
 - 3. Systém vyplní vstupné údaje pomocou validných dát získaných z XML File

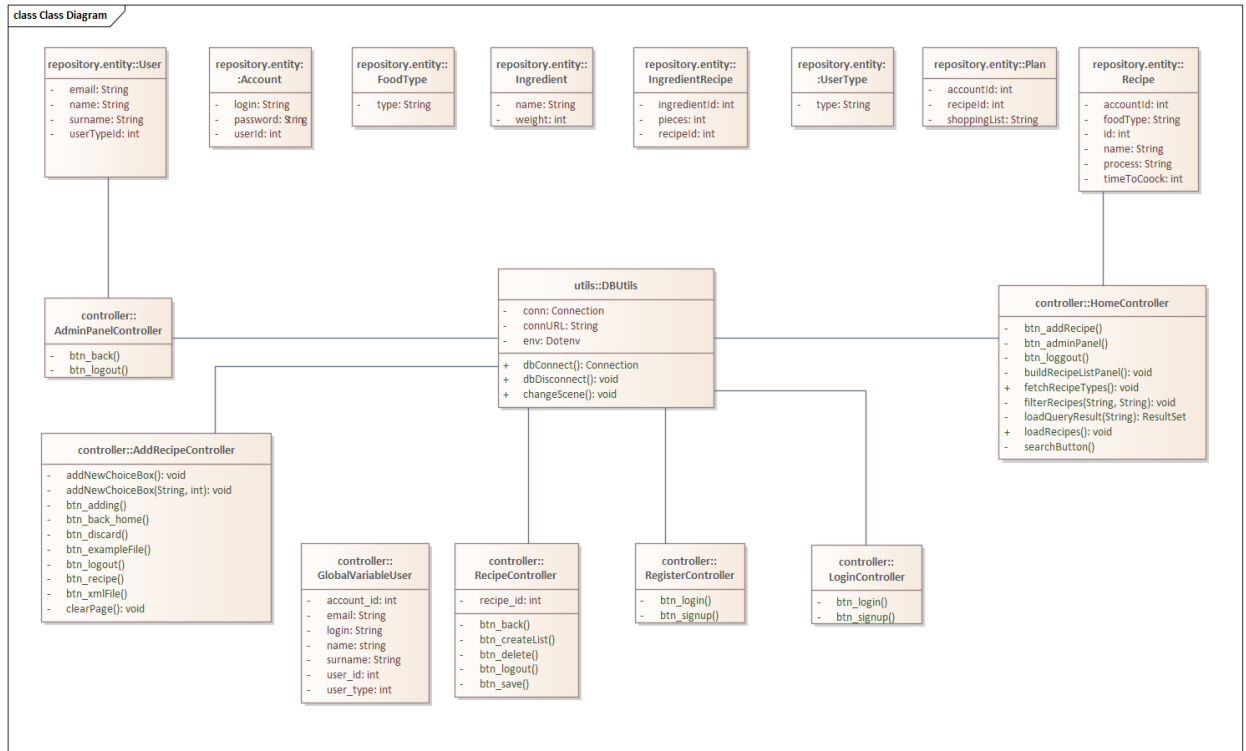
- 4. Používateľ má možnosť vstupné údaje editovať
 - 5. Používateľ klikne tlačidlo Uložiť
 - 6. Systém v prípade validných vstupných údajov uloží recept do databázy ako kompletný
 - 7. Systém v prípade neplatných údajov upozorní používateľa kde nastala chyba
- 2. Pridanie receptu krok po kroku
 - 1. Používateľ klikne na tlačidlo Vytvoriť nový recept
 - 2. Používateľ vyplní vstupné údaje
 - 3. Systém v prípade validných vstupných údajov uloží recept do databázy ako kompletný
 - 4. Systém v prípade neplatných údajov upozorní používateľa kde nastala chyba
- UC04 Vyhľadaj recept
 - 1. V hlavnej stránke zadaj kľúčové slovo pre želaný recept
 - 2. Klikni na tlačidlo "Hľadať"
- UC05 Uprav recept
- UC06 Zobraz recept
 - 1. Vykonaj UC04
 - 2. Kliknutím na recept zobraz želaný recept
- UC07: Vygeneruj nákupný zoznam
 - 1. Vykonaj UC04
 - 2. Vykonaj UC06
 - 3. Používateľ klikne na tlačidlo Vygeneruj nákupný zoznam
 - 4. Systém vygeneruje pre používateľa nákupný zoznam

8. Business model



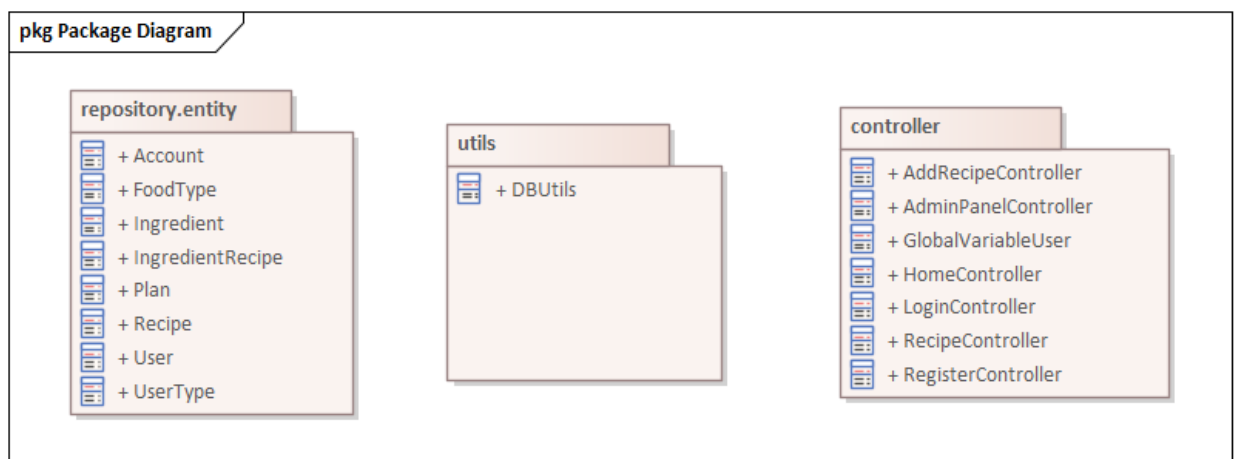
V princípe v našej aplikácii vieme zapísať a vymenovať množstvo business procesov, no zvolili sme si jeden taký najhlavnejší - Pridanie receptu. Ako vstupný business event máme záujem o pridanie receptu a výsledný business event recept je pridany. Ako business rolu máme kuchára, pretože pridávať recepty vedia práve kuchári/šéfkuchári, alebo admini. Vnútri business procesu máme 2 procesy - Pridanie obsahu receptu a uloženie receptu. Ako biznis objekty máme obsah receptu a samotný recept.

9. Class Diagram



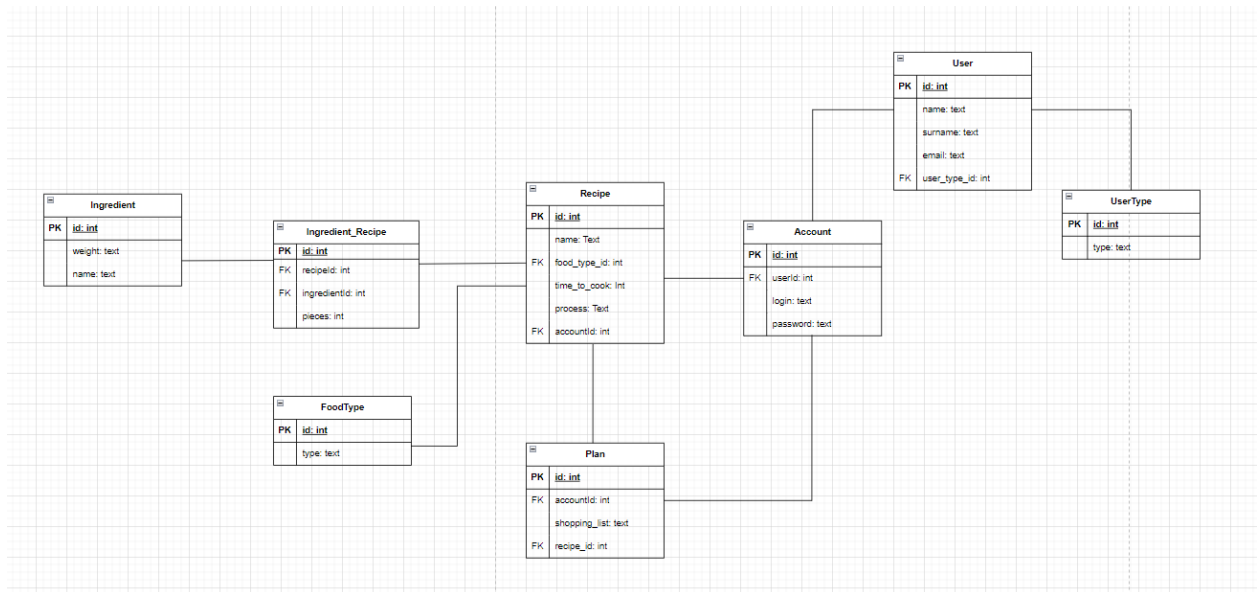
Pred implementáciou projektu sme navrhli class diagram, ktorý obsahuje všetky potrebné triedy, ktoré aplikácia bude obsahovať. Taktiež triedy budú mať všetky atribúty a tiež metódy, vrátane navrhnutých buttons.

10. Package Diagram



V package diagrame máme 3 package: repository.entity, controller, utils.

11. Databázový model



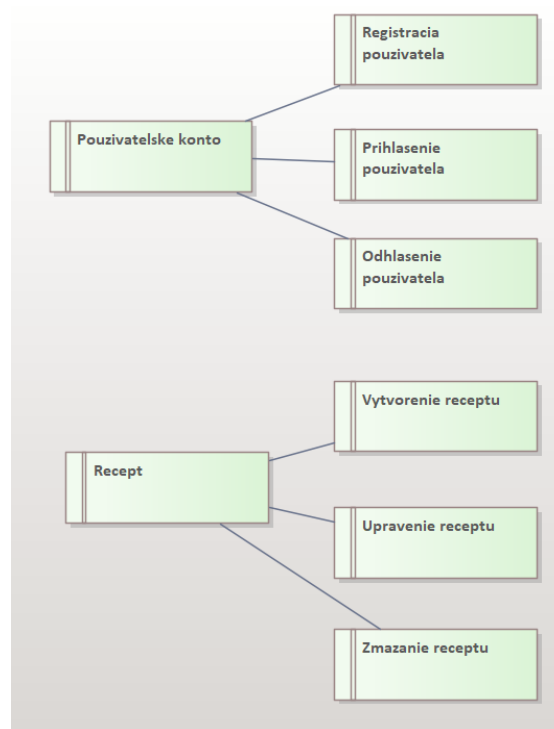
V našom databázovom modeli sú popísané nasledovné tabuľky:

- **Account**: tabuľka, ktorá obsahuje základné informačné údaje o používateľskom konte, ako sú login a heslo. Taktiež je táto tabuľka rozšírená o tabuľku User.
- **User**: v tejto tabuľke sú uložené kontaktné informácie uložené o používateľovi, a to: meno, priezvisko a email. Táto tabuľka je ešte rozšírená o tabuľku UserType.
- **UserType**: tabuľka, ktorá nám hovorí, o akého používateľa ide - či sa jedná o admina alebo o obvyčajného používateľa.
- **Recipe**: tabuľka, kde sú uložené recepty a to názov receptu (name), čas varenia (time_to_cook) a proces (process). Táto tabuľka Recipe je rozšírená o ďalšie 3 tabuľky: Plan, FoodType a Ingredient_Recipe
- **FoodType**: tabuľka, v ktorej sú uložené dáta o type jedla
- **Plan**: tabuľka, ktorá obsahuje a ukladá nákupný zoznam a takisto je viazaná na tabuľku Account a Recipe.
- **Ingredient_Recipe**: kontingenčná tabuľka, ktorá spája tabuľky Recipe a **Ingredient** ako reprezentácia many-to-many relácie a ukladá informácie o ingredienciách v recepte (weight a name - množstvo a názov).

12. Funkčné a nefunkčné požiadavky

Funkčné požiadavky:

- Pridanie receptu
- Úprava receptu
- Zmazanie receptu
- Prihlásenie sa do používateľského konta
- Registrácia nového používateľského konta



Nefunkčné požiadavky

- Lokalizácia
- Kompatibilita
- Škálovateľnosť
- Logovanie
- Dostupnosť



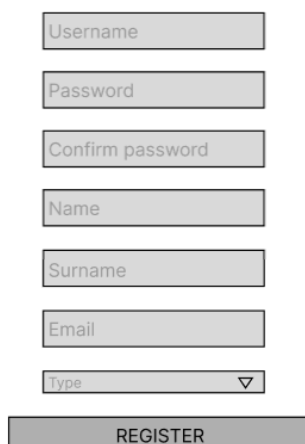
13. Základný návrh UX

Na základný návrh aplikácie, ako by mal vyzerat' design aplikácie, sa použil program FIGMA, kde sme namodelovali obrazovky našej aplikácie. Samozrejme, finálny produkt sa môže od nášho základného návrhu trochu líšiť.

CookManager

Prvý základný screen prihlasovania, kde používateľ zadá prihlasovacie meno a heslo a následne klikne na tlačidlo Login. Pre prípad, že používateľ ešte nemá vytvorené konto, klikne na tlačidlo Register.

CookManager



A vertical registration form with the following fields: Username, Password, Confirm password, Name, Surname, Email, and a dropdown menu for Type. Below these fields is a large REGISTER button.

Obrazovka registrácie, kde používateľ zadá svoje základné osobné údaje.



The main user interface features an orange header bar with the username 'xkovald' and three buttons: 'Admin Panel', 'Create new recipe', and 'Logout'. Below the header, there is a 'Vyfiltruj recept' dropdown menu and a search section with a 'Vyhľadaj recept' input field and a 'Hľadaj' button. Two recipe cards are displayed: 'Mac&Cheese' by Joseph Grey (Main foodtype, 30 mins) and 'Chicken Soup' by Joseph Black (Soup foodtype, 30 mins).

Hlavná obrazovka používateľa, kde sú zobrazené všetky recepty a kde je možné filtrovať recepty, vytvárať nové recepty, odhlásiť sa a kliknúť na svoje nastavenia.

xkovald
Back
Logout

Recipe name:
Ingredients:

Type of recipe:

Time to cook:

Steps:

XML FILE:

Obrazovka na pridanie receptu, kde používateľ zadá názov receptu, typ receptu, čas na varenie, ingrediencie a postup. Následne tlačidlom CREATE potvrdí vytvorenie receptu.

xkovald
Back
Logout

Recipe name:
Mac&Cheese

Type of recipe:
Pasta

Time to cook:
30 min

Ingredients:
Pasta
Eidam
Cheddar
Whipping Cream

Steps:
At first, you need to cook pasta. When a water boils, put pasta there and cook it for 10 minutes. Next put whipping cream to another pot, add grated cheese there and mix it with the spoon. Later mix this cream with pasta and put it into the pipe for 10 minutes for 200 *C.

Obrazovka na zobrazenie receptu, kde si používateľ môže prezerať recepty.

14. Načítanie pomocou XML

Aplikácia umožňuje vytváranie receptu viacerými spôsobmi a to sú načítanie z XML súboru, načítanie z XML súboru s úpravami, vyplnenie vstupných informácií používateľom.

Používateľské rozhranie pre pridanie receptov obsahuje tieto základné viditeľné atribúty.

ID	Atribút
1	Meno receptu
2	Typ jedla
3	Čas prípravy
4	Postup
5	Pridať ingredienciu
6	Ingrediencia
7	Predpríprava XML
8	Načítanie súboru XML
9	Zahodiť
10	Uložiť
11	Späť
12	Odhlásenie
13	Informačná hláška

Každé meno receptu je unikátne. Nemôže byť súčasne viac receptov s rovnakým názvom. Názov receptu je kontrolovaný pred tým než je recept pridaný do databázy.

Každé jedlo môže spadať práve do 1 z množstva predpripravených kategórií. Tento údaj je povinný.

Čas prípravy je povinný údaj ktorý vyjadruje čas potrebný na vykonanie všetkých úkonov receptu. Pre vstup je akceptovateľné iba číslo.

Kroky alebo postup varenia je povinný údaj ktorý má objasniť postup varenia, pečenia a prípravy jedál.

Tlačidlo pridať ingredienciu slúži na pridanie atribútu ingrediencia vďaka ktorému môžeme pridávať počet ingrediencií do receptu.

Ingrediencia sa skladá z 3 častí ,prvou časťou je výber ingrediencie z predpripravených ingrediencií. Druhou časťou je zadanie množstva koľko danej ingrediencie je potrebnej na vykonanie receptu táto hodnota musí byť zadaná číslom. Treťou časťou je tlačítko delete ktorou vymažeme zvolenú ingredienciu z receptu.

Tlačidlo predprípravy XML vytvorí súbor XML obsahujúci štruktúru akceptovateľného XML súboru do ktorého je možné doplniť údaje a následne načítať v aplikácii.

Tlačidlo zvoliť súbor XML otvorí súbor XML a v prípade že je to validný súbor pre aplikáciu vyplnia sa vstupné požiadavky pre vytvorenie receptu.

Tlačidlo zahodiť vymaže práve prebiehajúcu výrobu receptu a vráti používateľa do domovského okna

Tlačidlo uložiť skontroluje korektnosť vstupných parametrov a v prípade úspešnosti všetkých parametrov uloží nový recept.

Tlačidlo späť vráti používateľa do predchádzajúceho okna.

Tlačidlo odhlásenie odhlási používateľa.

Informačná hláška upozorňuje na validačné chyby vstupu.

15. Funkčnosť aplikácie

Po spustení aplikácie sa používateľ musí na začiatok prihlásiť pomocou prihlasovacieho mena a hesla.



Používateľské meno:

Heslo:

Prihlásenie

[Už zaregistrovaný?](#) [Registrovať](#)

Pokiaľ používateľ nemá žiadne používateľské konto, musí kliknúť na Registrovať, kde zadá základné informácie - Používateľské meno, heslo, potvrdenie hesla, Meno, Priezvisko, Email a typ používateľa (Guest alebo Chef).



Používateľské meno:

Heslo:

Potvrď heslo:

Meno:

Priezvisko:

Email:

Typ:

Zaregistruj sa!

[Už zaregistrovaný?](#) [Log In!](#)

Po úspešnom prihlásení sa používateľovi zobrazí hlavná stránka, kde sú zobrazené recepty. Ak je používateľ registrovaný ako Guest, tak nevie pridávať recepty, iba si ich pozerať kliknutím na daný recept, kde vidí názov receptu, typ jedla, čas na varenie, ingrediencie a kroky na prípravu. Takisto naľavo dole pri recepte je možnosť vygenerovať nákupný zoznam.

Vytvor nový recept

Odhlásiť

Vyhľadaj recept

Hľadaj

Palacinky
Food type: Dessert
Time to cook: 15 min.

Author: Unknown

rezen s kasou a slivkami
Food type: Vegan
Time to cook: 90 min.

Author: Unknown

rezen s kasou a hoj
Food type: Vegan
Time to cook: 90 min.

Author: Unknown

rezen s repou
Food type: Vegan
Time to cook: 90 min.

Author: Unknown

Spat

Odhlásiť

Meno receptu:
Typ jedla:
Čas prípravy:
Kroky:

Palacinky
Dessert
15

["key": "s lekvárom"]

Ingrediencie:

Vytvor nákupný zoznam

Zmazať

Uložiť

V možnostiach má používateľ filtrovať recept podľa typu jedla a hľadať recept podľa napríklad názvu jedla.

18

V prípade, ak je používateľ prihlásený ako Chef/Admin, má v hornom navbare možnosť pridať recept.

Po kliknutí na pridanie receptu sa používateľovi zobrazí obrazovka, kde zadá názov receptu, typ jedla, čas na varenie, vyberie si ingrediencie a napíše postup. Ak má vopred pripravený recept, môže ho importovať cez XML súbor. Po napísaní receptu je napravo vpravo tlačidlo na potvrdenie pridania alebo na zahodenie receptu.

The screenshot shows a web interface for creating a new recipe. At the top is an orange navigation bar with buttons for 'Späť' (Back) and 'Odhlásenie' (Logout). The main form is divided into several sections: 'Meno receptu' (Recipe name) with a text input field; 'Typ jedla:' (Food type) with a dropdown menu; 'Čas prípravy:' (Preparation time) with a text input field; and 'Kroky:' (Steps) with a large text area. To the right of these fields is a section for 'Ingrediencie:' (Ingredients) with a 'Pridať ingredienciu' (Add ingredient) button and a large empty box. At the bottom left, there is a 'Súbor XML:' section with buttons for 'Pridať XML súbor' and 'Vložiť súbor XML'. At the bottom right, there are buttons for 'Zahodiť' (Delete) and 'Uložiť' (Save).

Admin má takisto v možnostiach upraviť recept - po kliknutí na akýkoľvek recept sa mu zobrazí v nasledovnom tvare, kde môže upraviť akýkoľvek atribút:

The screenshot shows the web interface for editing an existing recipe. The layout is similar to the creation form, but with additional editing capabilities. The 'Meno receptu' field now contains the text 'test15'. The 'Typ jedla:' dropdown is set to 'Dessert'. The 'Čas prípravy:' field contains the number '32'. The 'Kroky:' text area contains the text 'hahd'. The 'Ingrediencie:' section now displays a list of ingredients with their quantities and unit of measurement, each with a 'Delete' button: 'Salt [g]' with a value of 100, 'Flour [g]' with a value of 50, and 'Beef meat [g]' with a value of 45. At the bottom right, there are buttons for 'Zahodiť' (Delete), 'Zmazať' (Erase), and 'Uložiť' (Save).

16. Dátová vrstva

Náš projekt je realizovaný v programovacom jazyku Java. Design sme riešili pomocou JavaFx. Databázu, ktorú používa naša aplikácia, beží na serveroch WebSupportu a jedná sa o databázu PostgreSQL.

- **Dependencies použité v projekte:**

- Javafx-controls 17.0.2
- Javafx-fxml 17.0.2
- Controlsfx 11.1.2
- Formsfx-core 11.5.0
- Bootstrapfx-core 0.4.0
- Junit-jupiter-api
- Junit-jupiter-engine
- PostgreSQL
- Java-dotenv 5.2.2
- Json 20180130
- Log4j-api 2.6.1
- Maven-compiler-plugin 3.10.1
- Javafx-maven-plugin 0.0.8

17. Login Controller

V tejto časti si popíšeme časti kódu - Login Controller.

```
@Override
public void initialize(URL url, ResourceBundle resourceBundle) {

    // xmacala +1
    btn_login.setOnAction(new EventHandler<ActionEvent>() {
        // xmacala +1
        @Override
        public void handle(ActionEvent actionEvent) {
            logger.info(String.format("User requested to log into account %s", tf_username.getText().trim()));

            boolean isAuthenticated = false;

            try {
                Connection conn = new DBUtils().dbConnect();

                String query = "SELECT * FROM public.account WHERE login=?";
                PreparedStatement pstmt = conn.prepareStatement(query);

                pstmt.setString(1, tf_username.getText().trim());
                ResultSet rs = pstmt.executeQuery();
```

Na začiatku sa nadviaže spojenie s databázou a hodnota isAuthenticated sa nastaví na hodnotu false a zavolá sa query, ktorá bude hľadať login.

```

while (rs.next()) {
    int user_id = rs.getInt( columnLabel: "user_id");
    String password = rs.getString( columnLabel: "password");
    if (password.equals(pf_password.getText())) {
        isAuthenticated = true;
        GlobalVariableUser.setUser(user_id, conn);
        DBUtils.dbDisconnect(conn);
        break;
    }
}
}

```

Následne sa v cykle zadajú údaje, až dokým sú zadané všetky údaje. Ak údaje sedia, hodnota isAuthenticated sa nastaví na hodnotu True a používateľ sa úspešne autentifikoval.

```

} catch (SQLException | NullPointerException e) {
    logger.error(e.getMessage());
}

if (isAuthenticated) {
    logger.info(String.format("User %s successfully logged in", GlobalVariableUser.getName()));
    DBUtils.changeScene(actionEvent, fxmlFile: "home.fxml", resourceBundle.getString( key: "cooking_manager"), resourceBundle);
} else {
    lbl_error.setText(resourceBundle.getString( key: "credentials_mismatch"));
    logger.error( s: "Incorrect credentials");
}
}
});

```

Ak je hodnota isAuthenticated nastavená na true (používateľ zadal úspešné údaje), zobrazí sa mu hlavná stránka. V opačnom prípade vyskočí hláška, že prihlasovacie údaje nie sú správne.

18. Home Controller

```

@Override
public void initialize(URL url, ResourceBundle resourceBundle) {
    new RecipeController();
    recipeScroll.setHbarPolicy(ScrollPane.ScrollBarPolicy.NEVER);
    recipeScroll.setStyle("-fx-background-color: transparent");
    lbl_name.setText(GlobalVariableUser.getName());
    searchButton.setStyle("-fx-background-color : #239c9c; -fx-background-radius : 15 15 15 15; -fx-text-fill: white;");
    searchButton.setOnMouseEntered(e -> searchButton.setStyle("-fx-background-color : #07f7f7; -fx-background-radius : 15 15 15 15; -fx-text-fill: black;"));
    searchButton.setOnMouseExited(e -> searchButton.setStyle("-fx-background-color : #239c9c; -fx-background-radius : 15 15 15 15; -fx-text-fill: white;"));
    if (GlobalVariableUser.getType() == 1){
        btn_addRecipe.setVisible(false);
        btn_adminPanel.setVisible(false);
    } else if (GlobalVariableUser.getType() == 2) {
        btn_adminPanel.setVisible(false);
    }
    loadRecipes(resourceBundle);
}

```

Na začiatku si inicializujeme všetky potrebné komponenty, ktoré sa nachádzajú v tomto controlleri, s ktorými hlavná stránka pracuje - napríklad filtrovanie receptu, pridávanie receptu.

```
public void loadRecipes(ResourceBundle resourceBundle) {
    DBUtils dbUtils = new DBUtils();
    HashMap<String, Recipe> recipes = new HashMap<>();

    try {
        Connection conn = dbUtils.dbConnect();

        String query = "SELECT r.id, r.name, r.account_id, r.time_to_cook, ft.type FROM recipe r" +
            " JOIN food_type ft ON ft.id = r.food_type_id";

        PreparedStatement pstmt = conn.prepareStatement(query);
        ResultSet rs = pstmt.executeQuery();

        // load data to hash-map for quicker search time
        while (rs.next()) {
            Recipe recipe = new Recipe();
            recipe.setId(rs.getInt( columnLabel: "id"));
            recipe.setName(rs.getString( columnLabel: "name"));
            recipe.setTimeToCook(rs.getInt( columnLabel: "time_to_cook"));
            recipe.setFoodType(rs.getString( columnLabel: "type"));

            recipes.put(rs.getString( columnLabel: "name"), recipe);
        }

        rs.close();
    }
}
```

Na načítanie receptov sa najskôr nadviaže spojenie s databázou, zavolá sa query na zobrazenie receptu, kde sa pripojí aj tabuľka food type. Na rýchlejšie spracovanie a vyhľadanie sa načítajú dáta do hash mapy.

```
// print recipes
for (String recipeKey : recipes.keySet()) {
    Recipe recipe = recipes.get(recipeKey);

    // get author of the recipe
    String accQuery = "SELECT u.\"name\", u.surname FROM account a" +
        " JOIN \"user\" u ON u.id=a.user_id" +
        " WHERE a.id=?";

    PreparedStatement pstmtAuthor = conn.prepareStatement(accQuery);
    pstmtAuthor.setInt( parameterIndex: 1, recipe.getAccountId());
    ResultSet rsAuthor = pstmtAuthor.executeQuery();

    // prepare presentation
    Pane recipePanel = new Pane();
    recipePanel.setStyle("-fx-background-color: #fff; -fx-padding: 20px; -fx-cursor: hand; -fx-border-radius: 15 15 15 15; -fx-border-color: #239c9c");

    // onClick event handler for recipes (opens recipe detail)
    recipePanel.setOnMouseClicked(e -> {
        RecipeController.setRecipe(recipe.getId());
        DBUtils.changeScene(e, fxmFile: "recipe.fxml", resourceBundle.getString( key: "cooking_manager"), resourceBundle);
    });
}
```

```

Text recipeName = new Text(recipe.getName());
recipeName.setStyle("-fx-font: normal bold 23px 'sans-serif'");
recipeName.setX(20.0);
recipeName.setY(40.0);

Text recipeType = new Text("Food type: " + recipe.getFoodType());
recipeType.setStyle("-fx-font: normal 18px 'sans-serif'");
recipeType.setX(20.0);
recipeType.setY(70.0);

Text timeToCook = new Text("Time to cook: " + recipe.getTimeToCook() + " min.");
timeToCook.setStyle("-fx-font: normal 18px 'sans-serif'");
timeToCook.setX(20.0);
timeToCook.setY(100.0);

Text author = new Text();
if (rsAuthor.next()) {
    author.setText("Author: " + rs.getString( columnLabel: "name") + " " + rs.getString( columnLabel: "surname"));
}
else {
    author.setText("Author: Unknown");
}

author.setStyle("-fx-font: normal 18px 'sans-serif'");
author.setX(750.0);
author.setY(40.0);

recipePanel.getChildren().addAll(recipeName, recipeType, timeToCook, author); // add component to the Pane component

recipeList.getChildren().add(recipePanel); // add new item to the VBox
recipeList.setStyle("-fx-background-color : white;");
}
}
catch (SQLException | NullPointerException e) {
    logger.error(e.getMessage());
}
}

```

Na výpis receptov sa v cykle vykonávajú všetky potrebné úkony, ktoré sú potrebné pri výpise receptu - zavolá sa query na získanie autora receptu. Po kliknutí na recept sa zavolajú potrebné metódy pre detaily o recepte (napríklad getName, getTimeToCook, getFoodType).

Admin Panel Controller je vypracovaný na podobnej báze, je tam len pridané pridanie receptu.

19. Recipe Controller

```
@Override
public void initialize(URL url, ResourceBundle resourceBundle) {

    Connection conn = new DBUtils().dbConnect();

    try {
        String query = "SELECT * FROM public.recipe WHERE id=?";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setInt( parameterIndex: 1, recipe_id);
        ResultSet rs = pstmt.executeQuery();

        while (rs.next()) {
            txt_recipeName.setText(rs.getString( columnLabel: "name").trim());
            txt_recipeTime.setText(String.valueOf(rs.getInt( columnLabel: "time_to_cook")).trim());
            txt_steps.setText(rs.getString( columnLabel: "process").trim());
            int food_type = rs.getInt( columnLabel: "food_type_id");

            query = "SELECT * FROM public.food_type WHERE id=?";
            pstmt = conn.prepareStatement(query);
            pstmt.setInt( parameterIndex: 1, food_type);
            rs = pstmt.executeQuery();

            while (rs.next()) {
                txt_recipeType.setText(rs.getString( columnLabel: "type").trim());
            }
        }
    } catch (SQLException e) {
        logger.error(e.getMessage());
    }
}
```

Na začiatok sa zavolá query, ktorá vyberie id receptu, nastaví sa tam podľa neho všetky potrebné hodnoty - názov, čas na varenie, process, food type na základe ďalšej query.

```
@Override
public void handle(ActionEvent actionEvent) {
    //vymazať recipe ako admin
    //recipe, ingredient_recipe, plan
    logger.info(String.format("Administrator requested to delete recipe number %d", recipe_id));
    try {
        String query = "SELECT * FROM public.plan WHERE recipe_id=?";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setInt( parameterIndex: 1, recipe_id);
        ResultSet rs = pstmt.executeQuery();

        while (rs.next()) {
            query = "DELETE FROM public.plan WHERE recipe_id=?";
            pstmt = conn.prepareStatement(query);
            pstmt.setInt( parameterIndex: 1, recipe_id);
            pstmt.executeUpdate();
        }

        query = "SELECT * FROM public.ingredient_recipe WHERE recipe_id=?";
        pstmt = conn.prepareStatement(query);
        pstmt.setInt( parameterIndex: 1, recipe_id);
        rs = pstmt.executeQuery();

        while (rs.next()) {
            query = "DELETE FROM public.ingredient_recipe WHERE recipe_id=?";
            pstmt = conn.prepareStatement(query);
            pstmt.setInt( parameterIndex: 1, recipe_id);
            pstmt.executeUpdate();
        }

        query = "DELETE FROM public.recipe WHERE id=?";
        pstmt = conn.prepareStatement(query);
        pstmt.setInt( parameterIndex: 1, recipe_id);
        pstmt.executeUpdate();
        DBUtils.changeScene(actionEvent, fxmlFile: "home.fxml", resourceBundle.getString( key: "cooking_manager"), resourceBundle);
    } catch (SQLException e) {
        logger.error(e.getMessage());
    }
}
```

Na zmazanie receptu sa opäť zavolá query, kde sa vyberie id receptu, ktorý sa ide zmazať. Následne sa prechádza v cykloch, kde sa volá DELETE query, kde

sa dosadí z rôznych tabuliek recipe_id. Po zmazaní receptu sa používateľovi zobrazí hlavná stránka. Na zmazanie receptu musí byť používateľ prihlásený ako chef/admin.

20. Add Recipe Controller

Ako prvý krok sa načítajú dáta z databázy pre scrollbar typu jedla a ingrediencií. ktoré sú následne doplnené ako options do týchto elementov.

```
try{//Get information about all food types and fill the panel
    Connection conn = new DBUtils().dbConnect();
    String query = "Select type from public.food_type";
    PreparedStatement pstmt = conn.prepareStatement(query);
    ResultSet rs = pstmt.executeQuery();
    while (rs.next()) {

        String food_type_element = rs.getString( columnLabel: "type");

        String arrNew[] = new String[arr.length + 1];
        int i;
        for(i = 0; i < arr.length; i++) {
            arrNew[i] = arr[i];
        }
        arrNew[i] = food_type_element;
        arr = arrNew.clone();
    }
    DBUtils.dbDisconnect(conn);
}
```

Po stlačení tlačidla na výber XML súboru sa otvorí zvolený súbor ktorý sa spracuje a podľa hodnôt z tohto súboru sa vložia hodnoti do príslušných okien.

```
try {
    // Create a new DocumentBuilderFactory and DocumentBuilder
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    DocumentBuilder builder = factory.newDocumentBuilder();
    // Parse the XML file into a Document object
    Document document = builder.parse(selectedFile.getAbsolutePath());
    // Get the root element of the XML document
    Element root = document.getDocumentElement();
    // Get a list of all the "recept" elements
    NodeList receptList = root.getElementsByTagName("recept");
    if (receptList.getLength() == 1){
        // Loop through each "recept" element and print its name, type, time, ingredients, and steps
        for (int i = 0; i < receptList.getLength(); i++) {
            org.w3c.dom.Node receptNode = receptList.item(i);
            if (receptNode.getNodeType() == org.w3c.dom.Node.ELEMENT_NODE) {
                Element receptElement = (Element) receptNode;
                String name = receptElement.getElementsByTagName("name").item( index: 0).getTextContent();
                String type = receptElement.getElementsByTagName("type").item( index: 0).getTextContent();
                String time = receptElement.getElementsByTagName("time").item( index: 0).getTextContent();
                NodeList ingredientList = receptElement.getElementsByTagName("ingredient");
                String steps = receptElement.getElementsByTagName("steps").item( index: 0).getTextContent();
            }
        }
    }
}
```

Pri stlačení tlačidla uložiť sa kontrolujú vložené údaje ako prvé sa kontroluje meno receptu pretože mená receptov sú jedinečné.

```
try{
    Connection conn = new DBUtils().dbConnect();

    String query = "SELECT name FROM public.recipe WHERE name=?";
    PreparedStatement pstmt = conn.prepareStatement(query);

    pstmt.setString(1, inputName.getText());
    ResultSet rs = pstmt.executeQuery();

    if (rs.next())
    {
        //System.out.println("This name is in DB");
        lbl_xmlFile.setText("Name already claimed");
        lbl_xmlFile.setTextFill(Color.color(1, 0, 0));
        DBUtils.dbDisconnect(conn);
    }
}
```

Každý recept má aspoň jednu ingredienciu, tieto ingrediencie sa uložia do poľa z ktorého budú neskôr posielané do databázy. V screene kódu môžeme vidieť čítanie child elementov Hbox nachádzajúcich sa v VBox. tieto Hboxy obsahujú názov ingrediencie, množstvo ingrediencie a možnosť vymazať ingredienciu.

```
ObservableList<Node> children = vbox_ingredients.getChildren();
List<List<Object>> myArray = new ArrayList<>();
int pom = 0;
for (Node child : children) {
    int data = 0;
    Object data2 = null;
    if (child instanceof HBox) {
        HBox hbox = (HBox) child;
        ObservableList<Node> hboxChildren = hbox.getChildren();
        for (Node hboxChild : hboxChildren) {
            if (hboxChild instanceof TextField) {
                TextField textField = (TextField) hboxChild;
                data = Integer.parseInt(textField.getText());
            } else if (hboxChild instanceof ChoiceBox<?>) {
                ChoiceBox choiceboxik = (ChoiceBox) hboxChild;
                data2 = choiceboxik.getValue();
            }
        }
    }
}
```

Dáta sú do databázy importované v 2 krokoch. V prvom kroku sa importuje do tabuľky recipe v druhom kroku sa importujú ingrediencie do tabuľky recipe_ingredient

```
String testing = new String( original: "INSERT INTO public.recipe (id, account_id, name, food_type_id, " +
    "time_to_cook, process) VALUES (DEFAULT, ?, ?, ?, ?, ?)");

if(myArray.size()>0){
try {

    conn = new DBUtils().dbConnect();

    pstmt = conn.prepareStatement(testing);
    pstmt.setInt( parameterIndex: 1, GlobalVariableUser.getUserId());
    pstmt.setString( parameterIndex: 2, inputName.getText());
    pstmt.setInt( parameterIndex: 3, type_id);
    pstmt.setInt( parameterIndex: 4, Integer.parseInt(inputTime.getText()));
    pstmt.setString( parameterIndex: 5, replacedString);
    pstmt.executeUpdate();
    conn.setAutoCommit(false); // start a transaction

    conn.commit(); // commit the transaction

    //System.out.println("All queries executed successfully");

    DBUtils.dbDisconnect(conn);
```

```
try {
    System.out.println("sem to padne");

    String recipe_testing = new String( original: "INSERT INTO public.ingredient_recipe (recipe_id, ingredient_id, pieces) VALUES (?, ?, ?)");
    //v myArray mám atm uložené System.out.println(myArray.get(au).get(0)); index 0 ingrediencia, 1množstvo, 2je ID
    conn = new DBUtils().dbConnect();
    pstmt = conn.prepareStatement(recipe_testing);

    for (int w = 0; w < myArray.size(); w++) {
        pstmt.setInt( parameterIndex: 1, recept_id);
        pstmt.setInt( parameterIndex: 2, (int) myArray.get(w).get(2));
        pstmt.setInt( parameterIndex: 3, (int) myArray.get(w).get(1));
        pstmt.addBatch();
    }

    pstmt.executeBatch();
```

Po rozkliknutí tlačidla pridať ingredienciu sa pridá element na obrazovku ktorý riadi správanie ingrediencie

```
btn_adding.setOnAction(event -> addNewChoiceBox());
```

21. UpdateRecipe Controller

Tento Controller slúži na zmenenie údajov v recepte. Zmeniť údaje daného receptu môže buď admin alebo majiteľ receptu.

```

try{
    Connection conn = new DBUtils().dbConnect();
    String query = "Select * from public.ingredient";
    PreparedStatement pstmt = conn.prepareStatement(query);
    ResultSet rs = pstmt.executeQuery();

    while (rs.next()) {
        int ing_id = rs.getInt( columnLabel: "id");
        String ing_name = rs.getString( columnLabel: "name");
        String ing_uom = rs.getString( columnLabel: "uom");

        int ing_weight = 0;
        String[] newArrayName = Arrays.copyOf(ingredientNameArray, newLength: ingredientNameArray.length + 1);
        String[] newArrayNameClear = Arrays.copyOf(ingredientNameArrayClear, newLength: ingredientNameArrayClear.length + 1);

        int k;
        for(k = 0; k < ingredientNameArray.length; k++) {
            newArrayName[k] = ingredientNameArray[k];
            newArrayNameClear[k] = ingredientNameArrayClear[k];
        }
        newArrayName[k] = ing_name + " [" + ing_uom + "]" ;
        newArrayNameClear[k] = ing_name ;

        ingredientNameArray = newArrayName.clone();
        ingredientNameArrayClear = newArrayNameClear.clone();

        Object[][] arrNew = new Object[arr_ing.length + 1][3];

        for (int i = 0; i < arr_ing.length; i++) {
            for (int j = 0; j < 3; j++) {
                arrNew[i][j] = arr_ing[i][j];
            }
        }

        arrNew[arrNew.length -1] = new Object[] {ing_id, ing_name, ing_uom, ing_weight};
        arr_ing = arrNew.clone();
    }
    DBUtils.dbDisconnect(conn);
}

```

V tomto prvom try-catch si do poľa Stringov uložíme všetky ingrediencie z databázy pomocou SELECTu.

```

try{
    Connection conn = new DBUtils().dbConnect();
    String query = "Select type from public.food_type";
    PreparedStatement pstmt = conn.prepareStatement(query);
    ResultSet rs = pstmt.executeQuery();

    //robíš že hľadáš types s DB
    while (rs.next()) {

        String food_type_element = rs.getString( columnLabel: "type");

        String arrNew[] = new String[arr.length + 1];
        int i;
        for(i = 0; i < arr.length; i++) {
            arrNew[i] = arr[i];
        }
        arrNew[i] = food_type_element;
        arr = arrNew.clone();
    }
    DBUtils.dbDisconnect(conn);

    //System.out.println(arr[1]);
    choiceType.setItems(FXCollections.observableArrayList(arr)
    );
}

```

V tomto try-catch si do poľa uložíme všetky typy jedál z databázy.

```

try {
    Connection conn = new DBUtils().dbConnect();
    String query = "Select process from public.recipe WHERE id=(?)";
    PreparedStatement pstmt = conn.prepareStatement(query);
    pstmt.setInt( parameterIndex: 1, recipe.getId());
    ResultSet rs = pstmt.executeQuery();

    while (rs.next()){
        inputSteps.setText(rs.getString( columnLabel: "process"));
    }
    DBUtils.dbDisconnect(conn);
} catch (SQLException e) {
    throw new RuntimeException(e);
}

```

V tomto try-catch si do Text Area komponentu vložíme proces receptu pripravený na možné upravovanie.

```

try {
    Connection conn = new DBUtils().dbConnect();
    String query = "SELECT i.name, i.uom, ir.pieces FROM ingredient i" +
        " JOIN ingredient_recipe ir ON i.id=ir.ingredient_id" +
        " WHERE ir.recipe_id=(?)";
    PreparedStatement pstmt = conn.prepareStatement(query);
    pstmt.setInt( parameterIndex: 1, recipe.getId());
    ResultSet rs = pstmt.executeQuery();

    while (rs.next()){
        addNewChoiceBox( ing_name: rs.getString( columnLabel: "name") + " [" + rs.getString( columnLabel: "uom") + "]", rs.getString( columnLabel: "pieces"));
    }

    DBUtils.dbDisconnect(conn);
}

```

V tomto try-catch si pomocou SELECTu a JOINu a funkcie ktú opíšem nižšie pridám do VBox komponentu všetky ingrediencie daného receptu.

```

    @Override
    public void handle(ActionEvent actionEvent) {
        logger.info("User logged out of the application");
        DBUtils.changeScene(actionEvent, fxmlFile: "login.fxml", resourceBundle.getString(key: "login_title"), resourceBundle);
    }
});

    @Override
    public void handle(ActionEvent actionEvent) {
        logger.info("User requested to go back to the home screen");
        DBUtils.changeScene(actionEvent, fxmlFile: "home.fxml", resourceBundle.getString(key: "cooking_manager"), resourceBundle);
    }
});

    @Override
    public void handle(ActionEvent actionEvent) {
        logger.info("User discarded recipe creation");
        DBUtils.changeScene(actionEvent, fxmlFile: "home.fxml", title: "Cooking Manager", resourceBundle);
    }
});

```

Na tomto obrázku sa nachádzajú 3 tlačidlá. Prvé je na odhlásenie, druhé na prepnutie späť na domovskú obrazovku a tretie taktiež vráti na domovskú obrazovku.

```

1 xmacala
btn_save.setOnAction(new EventHandler<ActionEvent>() {
    1 xmacala
    @Override
    public void handle(ActionEvent actionEvent) {
        try {
            Connection conn = new DBUtils().dbConnect();
            String query = "UPDATE public.recipe SET account_id=? WHERE id=?";
            PreparedStatement pstmt = conn.prepareStatement(query);
            pstmt.setInt( parameterIndex: 1, GlobalVariableUser.getAccountId());
            pstmt.setInt( parameterIndex: 2, recipe.getId());
            pstmt.executeUpdate();

            query = "UPDATE public.recipe SET `name`=? WHERE id=?";
            pstmt = conn.prepareStatement(query);
            pstmt.setString( parameterIndex: 1, inputName.getText());
            pstmt.setInt( parameterIndex: 2, recipe.getId());
            pstmt.executeUpdate();

            query = "UPDATE public.recipe SET time_to_cook=? WHERE id=?";
            pstmt = conn.prepareStatement(query);
            pstmt.setInt( parameterIndex: 1, Integer.parseInt(inputTime.getText()));
            pstmt.setInt( parameterIndex: 2, recipe.getId());
            pstmt.executeUpdate();

            query = "UPDATE public.recipe SET process=? WHERE id=?";
            pstmt = conn.prepareStatement(query);
            pstmt.setString( parameterIndex: 1, inputSteps.getText());
            pstmt.setInt( parameterIndex: 2, recipe.getId());
            pstmt.executeUpdate();

            query = "SELECT * FROM public.food_type WHERE type=?";
            pstmt = conn.prepareStatement(query);
            pstmt.setString( parameterIndex: 1, (String) choiceType.getValue());
            ResultSet rs = pstmt.executeQuery();

            while (rs.next()) {
                int id = rs.getInt( columnLabel: "id");
                query = "UPDATE public.recipe SET food_type_id=? WHERE id=?";
                pstmt = conn.prepareStatement(query);
                pstmt.setInt( parameterIndex: 1, id);
                pstmt.setInt( parameterIndex: 2, recipe.getId());
                pstmt.executeUpdate();
            }
            DBUtils.changeScene(actionEvent, fxmlFile: "home.fxml", title: "Cooking Manager", resourceBundle);
        }
    }
}

```

Po kliknutí na toto tlačidlo btn_save sa nám aktualizujú všetky údaje ktoré používateľ upravil alebo aj tie ktoré nechal tak.

```

btn_delete.setOnAction(new EventHandler<ActionEvent>() {
    1 xmacala
    @Override
    public void handle(ActionEvent actionEvent) {
        //vymazať recipe ako admin
        //recipe, ingredient_recipe, plan
        Logger.info(String.format("Administrator requested to delete recipe number %d", recipe_id));
        try {
            Connection conn = new DBUtils().dbConnect();
            String query = "SELECT * FROM public.plan WHERE recipe_id=?";
            PreparedStatement pstmt = conn.prepareStatement(query);
            pstmt.setInt( parameterIndex: 1, recipe_id);
            ResultSet rs = pstmt.executeQuery();

            while (rs.next()) {
                query = "DELETE FROM public.plan WHERE recipe_id=?";
                pstmt = conn.prepareStatement(query);
                pstmt.setInt( parameterIndex: 1, recipe_id);
                pstmt.executeUpdate();
            }

            query = "SELECT * FROM public.ingredient_recipe WHERE recipe_id=?";
            pstmt = conn.prepareStatement(query);
            pstmt.setInt( parameterIndex: 1, recipe_id);
            rs = pstmt.executeQuery();

            while (rs.next()) {
                query = "DELETE FROM public.ingredient_recipe WHERE recipe_id=?";
                pstmt = conn.prepareStatement(query);
                pstmt.setInt( parameterIndex: 1, recipe_id);
                pstmt.executeUpdate();
            }

            query = "DELETE FROM public.recipe WHERE id=?";
            pstmt = conn.prepareStatement(query);
            pstmt.setInt( parameterIndex: 1, recipe_id);
            pstmt.executeUpdate();
            DBUtils.changeScene(actionEvent, fxmlFile: "home.fxml", resourceBundle.getString( key: "cooking_manager"), resourceBundle);
        }
    }
}

```

Po kliknutí na toto tlačidlo btn_delete vymaže všetky potrebné údaje z databázy ktoré sa týkajú tohto receptu.

```
private void addNewChoiceBox(String ing_name, String amount) {

    HBox hbox = new HBox();
    hbox.setSpacing(10);

    ChoiceBox<String> choiceBox = new ChoiceBox<>();
    choiceBox.getItems().addAll(ingredientNameArray);
    choiceBox.setValue(ing_name);

    Button deleteButton = new Button("Delete");
    deleteButton.setOnAction(event -> {
        vbox_ingredients.getChildren().remove(hbox);
    });

    TextField textField = new TextField();
    textField.setText(amount);
    textField.setPrefWidth(45);

    hbox.getChildren().addAll(choiceBox, textField, deleteButton);

    vbox_ingredients.getChildren().add(hbox);
}
```

V tejto funkcii si do VBoxu pridáme všetky ingrediencie daného receptu ktoré už obsahuje.

Login.fxml

V FXML súbore login.fxml sme použili na design TextFieldy na username, passwordField na heslo, Button na tlačidlo prihlásenia a ImageView na logo.


```

<AnchorPane prefHeight="120.0" prefWidth="120.0" style="-fx-background-color: white;" xmlns="http://javafx.com/javafx/9" xmlns:fx="http://javafx.com/fxml/1" fx:controller="eu.jlll.cookingmanager.cookingmanager.controller.LoginController">
    <Label layoutX="400.0" layoutY="110.0" text="Username">
        <font>
            <font name="System Bold Italic" size="18.0" />
        </font>
    </Label>
    <Label layoutX="400.0" layoutY="166.0" text="Pass_login">
        <font>
            <font name="System Bold Italic" size="18.0" />
        </font>
    </Label>
    <TextField fx:id="tf_username" layoutX="400.0" layoutY="110.0" promptText="Username_field">
        <font>
            <font size="14.0" />
        </font>
    </TextField>
    <Button fx:id="btn_login" layoutX="600.0" layoutY="400.0" mnemonicParsing="false" prefHeight="43.0" prefWidth="134.0" style="-fx-cursor: hand;" styleheets="@../css/style.css" text="Kloga">
        <font>
            <font name="System Bold Italic" size="14.0" />
        </font>
    </Button>
    <Button fx:id="btn_signup" layoutX="700.0" layoutY="507.0" mnemonicParsing="false" style="-fx-background-color: transparent;" style="-fx-cursor: hand;" text="Najnu" textFill="#000000">
    </Button>
    <Label layoutX="570.0" layoutY="400.0" text="Nalrady_reg">
        <font>
            <font name="System Italic" size="14.0" />
        </font>
    </Label>
    <PasswordField fx:id="pf_password" layoutX="400.0" layoutY="166.0" promptText="Password_field">
        <font>
            <font size="14.0" />
        </font>
    </PasswordField>
    <Image fx:id="img_logo" layoutX="400.0" layoutY="110.0" pickOnBounds="true" preserveRatio="true">
        <image>
            <image url="@../assets/recipebook_logo.png" />
        </image>
    </Image>
    <Label fx:id="lbl_error" alignment="CENTER" contentDisplay="CENTER" layoutX="500.0" layoutY="166.0" textFill="RED">
        <font>
            <font name="System Italic" size="13.0" />
        </font>
    </Label>
</AnchorPane>

```

22. Register.fxml

```

<AnchorPane prefHeight="120.0" prefWidth="120.0" style="-fx-background-color: white;" xmlns="http://javafx.com/javafx/9" xmlns:fx="http://javafx.com/fxml/1" fx:controller="eu.jlll.cookingmanager.cookingmanager.controller.RegisterController">
    <Label layoutX="400.0" layoutY="110.0" text="Username">
        <font>
            <font name="System Bold Italic" size="18.0" />
        </font>
    </Label>
    <Label layoutX="400.0" layoutY="166.0" text="Pass_login">
        <font>
            <font name="System Bold Italic" size="18.0" />
        </font>
    </Label>
    <TextField fx:id="tf_username" layoutX="400.0" layoutY="110.0" promptText="Username_field">
        <font>
            <font size="14.0" />
        </font>
    </TextField>
    <Button fx:id="btn_signup" layoutX="600.0" layoutY="400.0" mnemonicParsing="false" style="-fx-cursor: hand;" styleheets="@../css/style.css" text="Najnu">
        <font>
            <font name="System Bold Italic" size="14.0" />
        </font>
    </Button>
</AnchorPane>

```

V register.fxml súbore využívame takisto Textfielidy na zadávanie osobných údajov používateľa, passwordField na heslo, Button na tlačidlo registrácie a ImageView na načítanie loga.

25. Záver

Pri vývoji aplikácie Cooking Manager sme sa na začiatok zamerali na správne rozdelenie úloh v našom tíme, architekti navrhli základné diagramy, podľa ktorých následne vytvorili backend a frontend developeri aplikáciu.

Aplikácia je zameraná na manažovanie receptov, pridávanie receptov s cieľom vymieňať si recepty medzi ľuďmi, naučiť sa variť, jednoducho spravovať bázu svojich receptov, atď.

V našom tíme sme sa naučili vzájomnej komunikácii počas tvorby celého projektu, spolupráci medzi pozíciami - napríklad spoluprácu architekta s backend developermi.

Takisto sme sa museli spoľahnúť jeden na druhého, že každý svoju prácu splní v dostatočnom predstihu, aby sme sa vyhli časovej tiesni.

Veríme, že naša aplikácia bude pre používateľov užitočná a budú ju využívať.