

Dokumentace k třetímu projektu BDS

Vypracoval:

Filip Texl

Brno 2021

Obsah

| | |
|--------------------------------------|----|
| 1. Úvod | 3 |
| 2. Popis aplikace | 3 |
| 3. Ukázka funkčnosti aplikace | 3 |
| Autentizace uživatele | 3 |
| Databáze zaměstnanců..... | 5 |
| 4. Splnění požadavky ze zadání | 11 |

1. Úvod

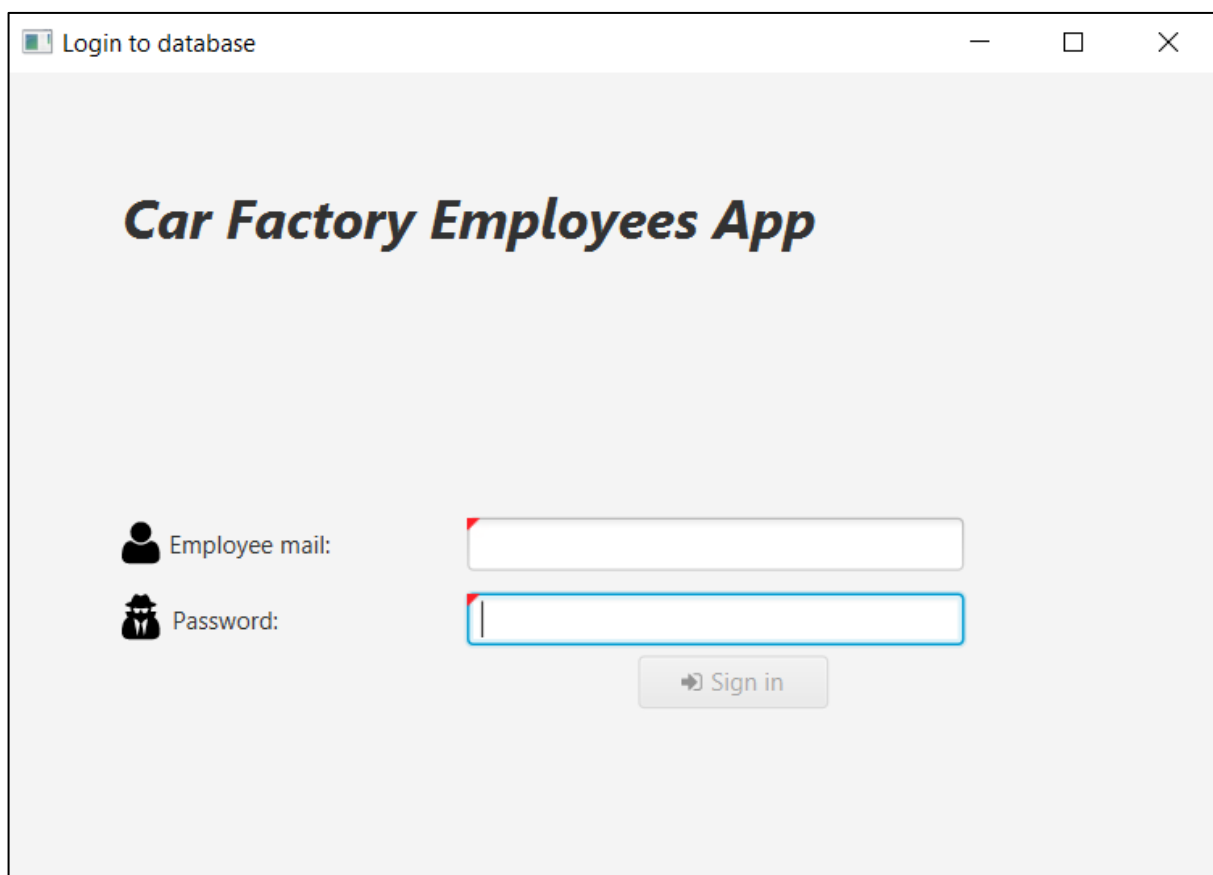
Cílem třetího projektu bylo vytvořit aplikaci v JAVĚ s GUI, která komunikuje s databází a provádí příslušné úkony. Projekt navazuje na předchozí projekty, kdy v prvním projektu jsme databázi navrhli a následně implementovali a v druhém projektu jsme se dotazovali nad touto databází.

2. Popis aplikace

Aplikace je navrhnutá pro správu zaměstnanců automobilky. Do databáze se přihlašuje zaměstnanec svým emailem a heslem z tabulky employee. Následně aplikace pomocí BCryptu autentizuje uživatele a zobrazí mu databázi zaměstnanců. Zde si může uživatel zobrazit detailní informace o zaměstnanci, či zaměstnance upravit. Je zde také možnost řazení sloupců ať už podle jména, či jiných atributů. V dolní části hlavního okna aplikace je také možnost databázi refreshnout, nebo zaměstnance přidat.

3. Ukázka funkčnosti aplikace

Autentizace uživatele



Login to database

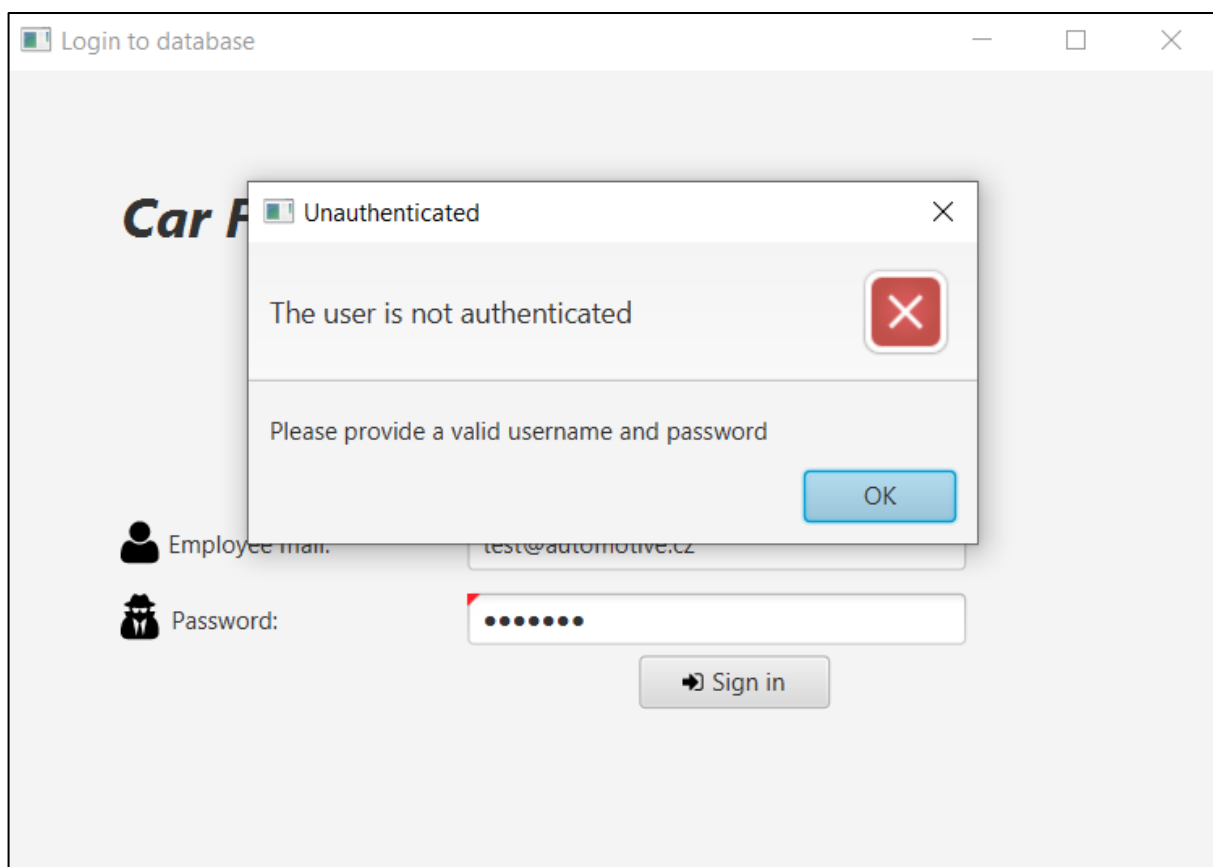
Car Factory Employees App

Employee mail:

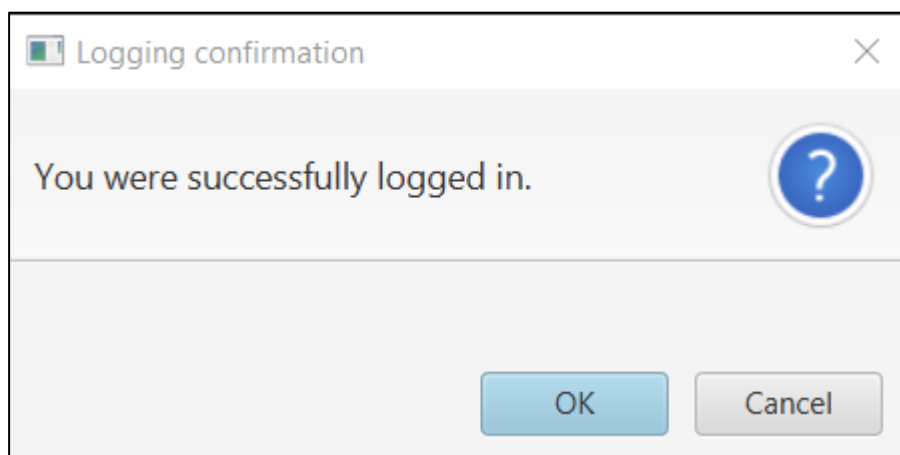
Password:

➔ Sign in

Obr. 1

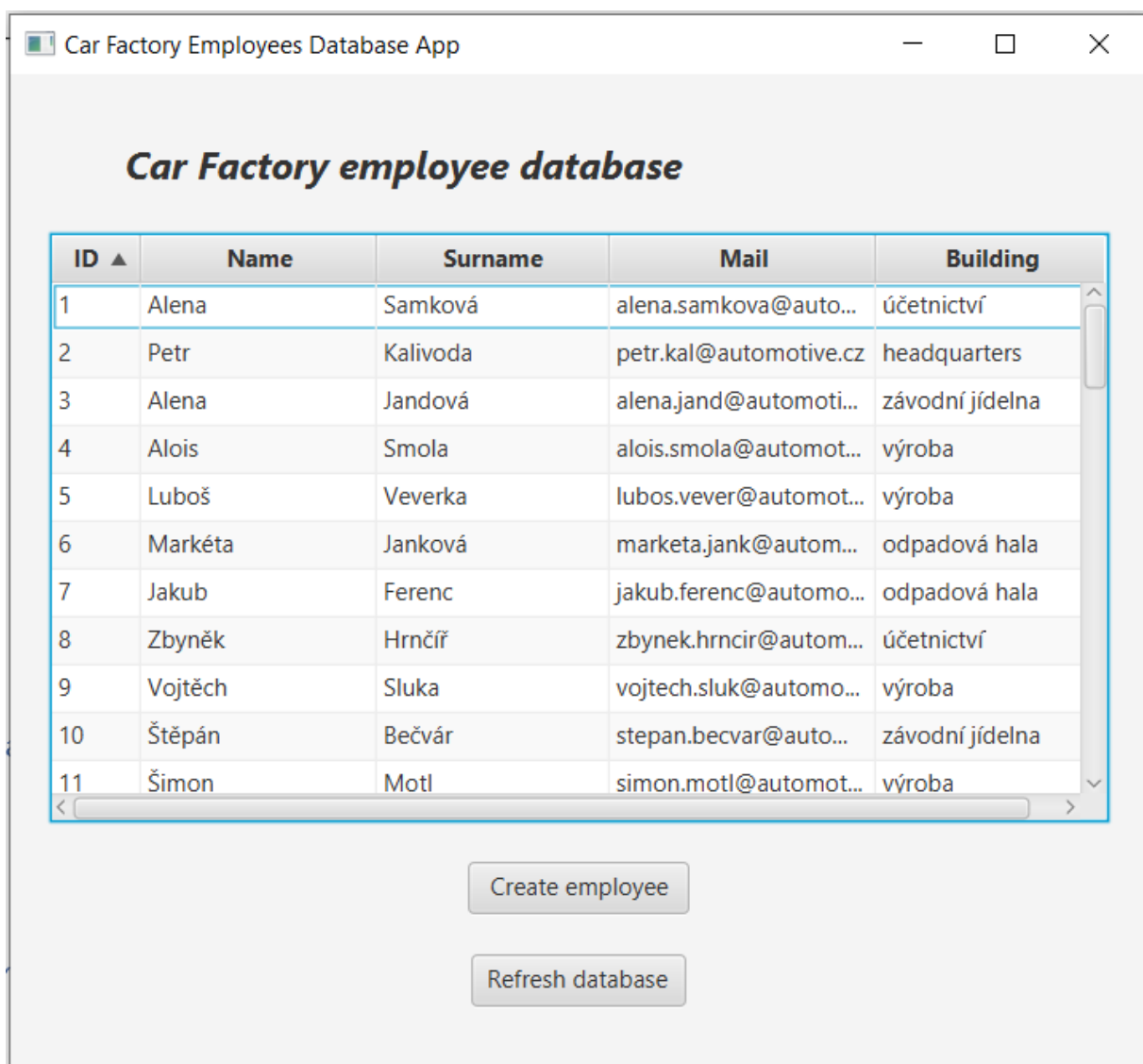


Obr. 2 - při nesprávném loginu či hesle uživatel není autentizován



Obr. 3 - při správném loginu a hesle je uživatel autentizován

Databáze zaměstnanců



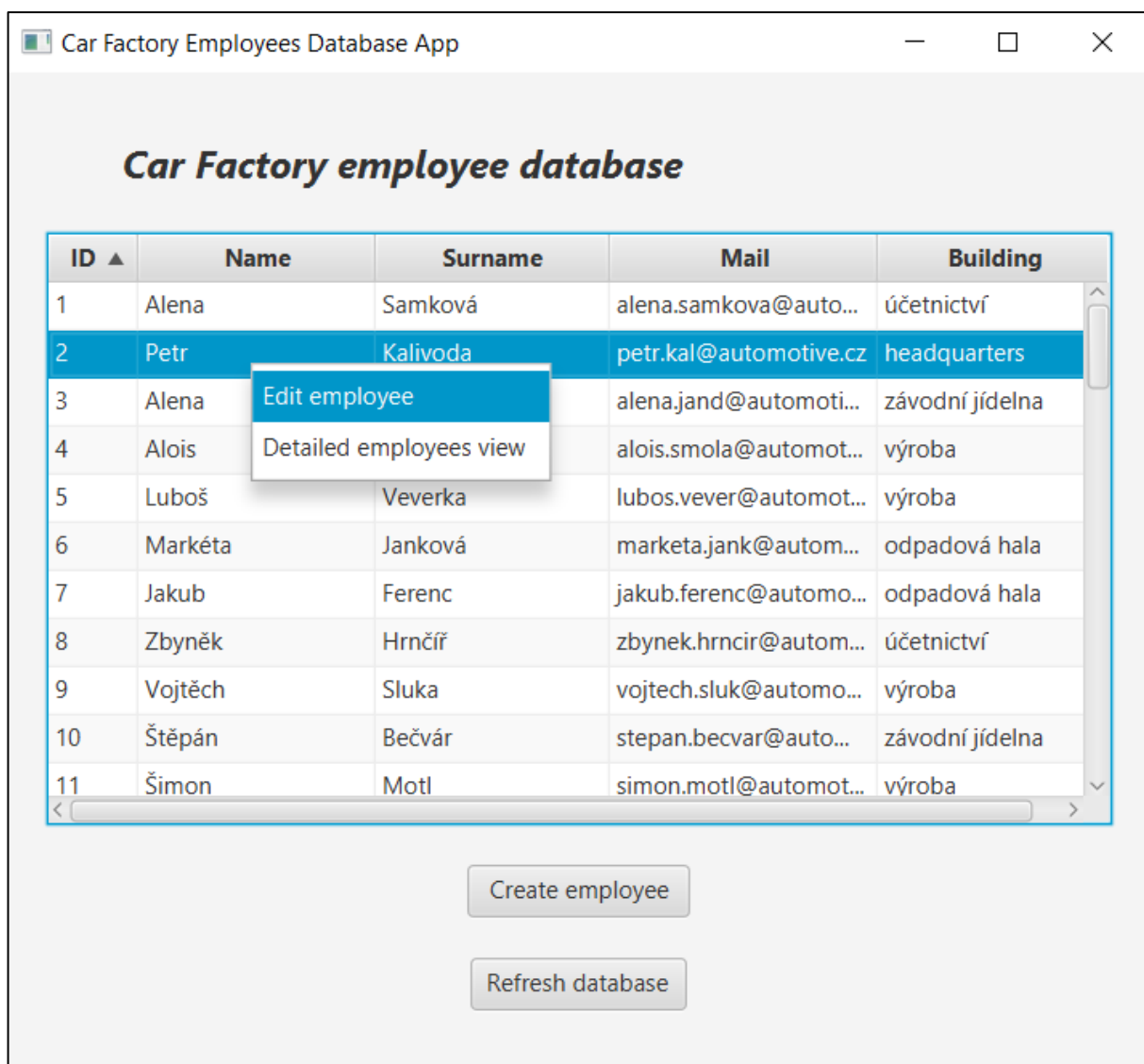
| ID ▲ | Name | Surname | Mail | Building |
|------|---------|----------|------------------------|-----------------|
| 1 | Alena | Samková | alena.samkova@auto... | účetnictví |
| 2 | Petr | Kalivoda | petr.kal@automotive.cz | headquarters |
| 3 | Alena | Jandová | alena.jand@automoti... | závodní jídelna |
| 4 | Alois | Smola | alois.smola@automot... | výroba |
| 5 | Luboš | Veverka | lubos.vever@automot... | výroba |
| 6 | Markéta | Janková | marketa.jank@autom... | odpadová hala |
| 7 | Jakub | Ferenc | jakub.ferenc@automo... | odpadová hala |
| 8 | Zbyněk | Hrnčář | zbynek.hrncir@autom... | účetnictví |
| 9 | Vojtěch | Sluka | vojtech.sluk@automo... | výroba |
| 10 | Štěpán | Bečvár | stepan.becvar@auto... | závodní jídelna |
| 11 | Šimon | Motl | simon.motl@automot... | výroba |

Create employee

Refresh database

Obr. 4 - po přihlášení se nám zobrazí databáze zaměstnanců

V základním pohledu můžeme vidět ID zaměstnance, jméno, příjmení, mail a budovu, ve které zaměstnanec pracuje. Nutno podotknout, že budova není v základní tabulce zaměstnance v databázi a i zde už byl použit JOIN.

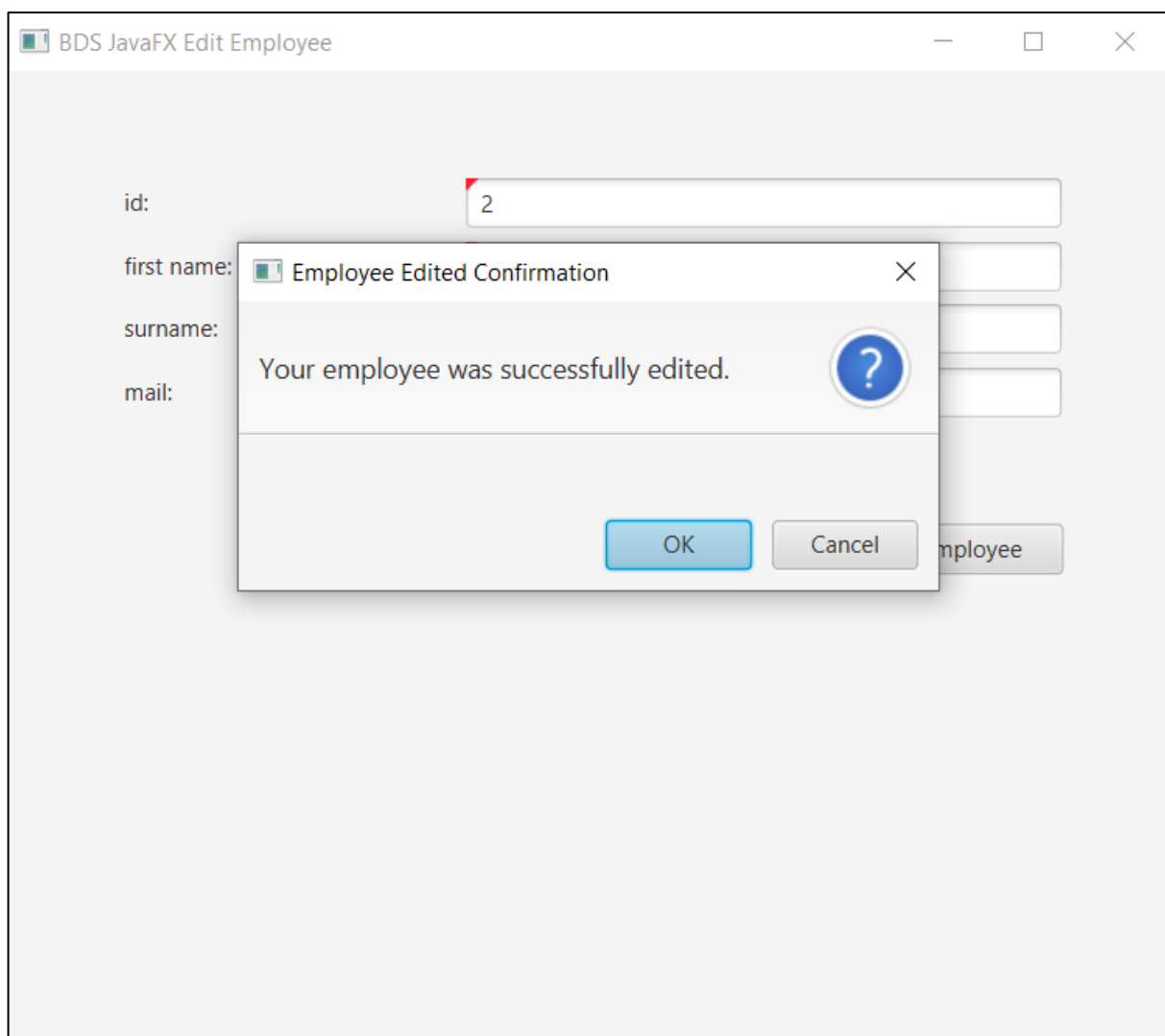


Obr. 5 - kliknutím pravým tlačítkem myši se nám zobrazí další dvě možnosti: detailní informace o zaměstnanci a možnost zaměstnance upravit

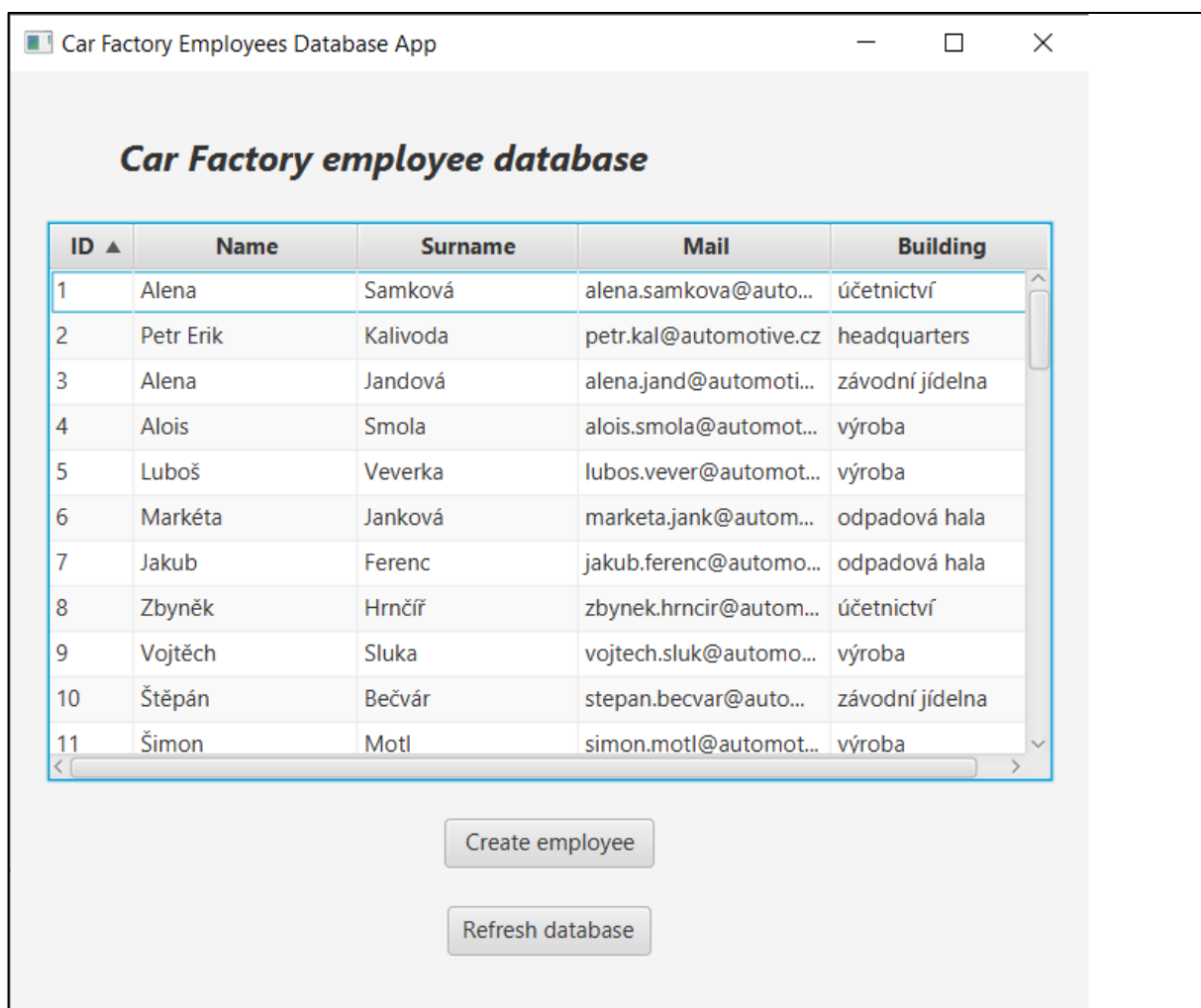
Employees Detailed View

| | |
|---------------|---|
| id: | <input type="text" value="2"/> |
| First name: | <input type="text" value="Petr"/> |
| Surname: | <input type="text" value="Kalivoda"/> |
| mail: | <input type="text" value="petr.kal@automotive.cz"/> |
| building: | <input type="text" value="headquarters"/> |
| job: | <input type="text" value="účetní"/> |
| city: | <input type="text" value="Zajecov"/> |
| street: | <input type="text" value="K Lukárně"/> |
| house number: | <input type="text" value="5188"/> |

Obr. 6 – detailní pohled zaměstnance, textová pole nejsou editovatelná



Obr. 7 – možnost zaměstnance upravit, po upravení nás informuje dialogové okno o úspěšném upravení zaměstnance



Obr. 8 – databáze po úpravě, musí se nejdříve stlačit tlačítko „Refresh database“

BDS JavaFX Create Employee

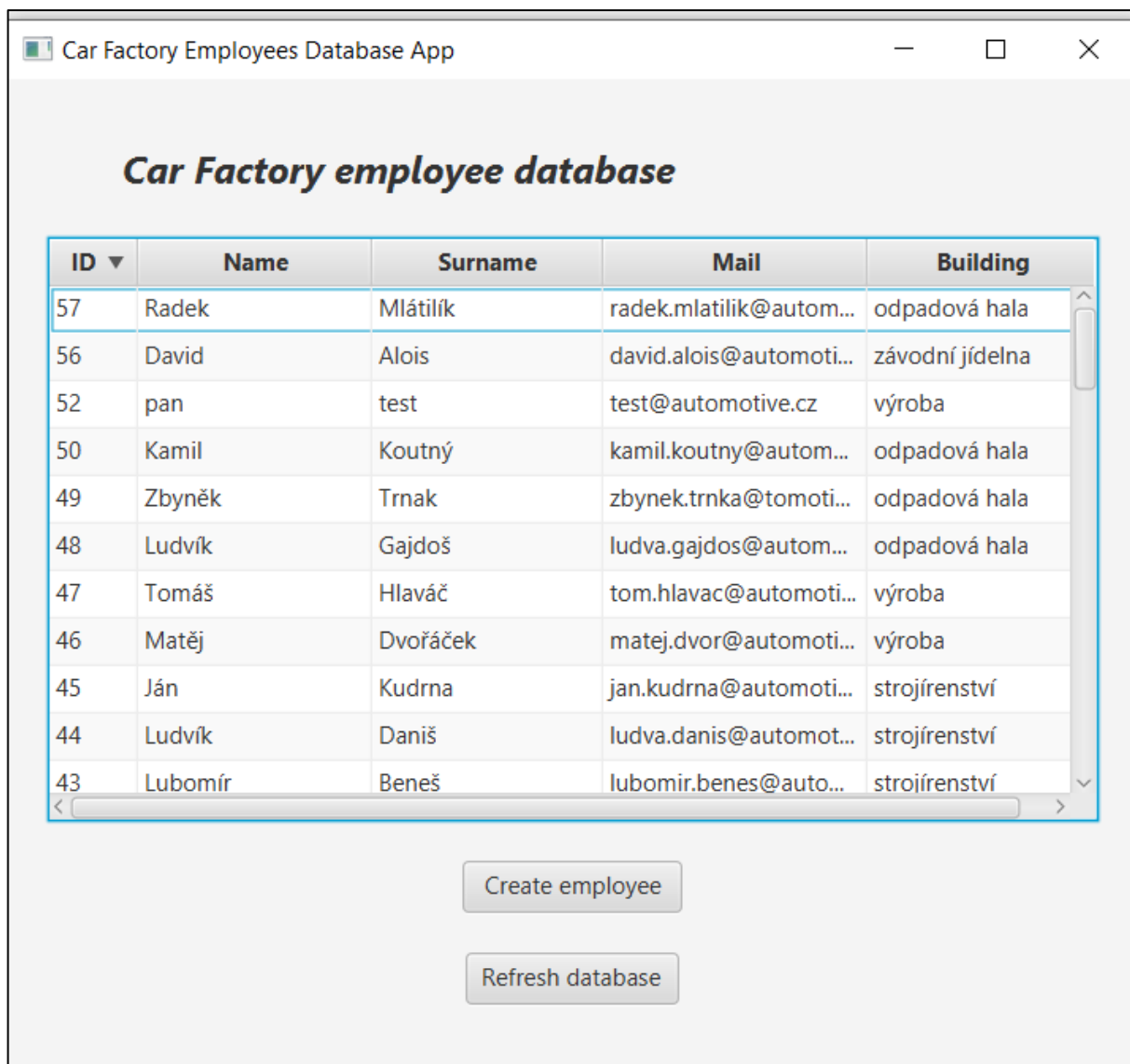
first name:

surname:

mail:

password:

Obr. 9 – možnost vytvoření zaměstnance



Obr. 10 – databáze zaměstnanců po přidání Radka Mlátílika

| | | | | | | |
|----|----|-------|----------|------------------------------|--|---|
| 53 | 57 | Radek | Mlátílik | radek.mlatilik@automotive.cz | \$2a\$12\$.dvHDedFWelcq0DZMULE.ZE50eNZa.C.514uyz0Py/K5bIHTW0gm | 5 |
|----|----|-------|----------|------------------------------|--|---|

Obr. 11 – změna se projeví i v databázi

4. Splněné požadavky ze zadání

You have to implement a Desktop application in Java or Python. For Java use Swing (taught in BPC-PC2T) or JavaFX, for Python, it is up to you. If you already know some web-based framework Spring/Jakarta EE/Django/Flask and arbitrary JavaScript library/framework you can also use it.

Splněno viz kapitola 3.

The application must be compilable and runnable from command-line (use Maven or Gradle – check seminar project), e.g., to compile/build mvn clean package and to run java -jar my-bds-app.jar.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.1415]
(c) Microsoft Corporation. Všechna práva vyhrazena.

C:\VUT\2. ročník ZS\BPC-BDS\3.projekt\project-demo2\BDSprojekt3>mvn clean install
[INFO] Scanning for projects...
[INFO] -----< org.but:project-demo >-----
[INFO] Building project-demo 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ project-demo ---
[INFO] Deleting C:\VUT\2. ročník ZS\BPC-BDS\3.projekt\project-demo2\BDSprojekt3\target
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ project-demo ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 7 resources
[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ project-demo ---
[WARNING] * Required filename-based automodules detected: [bcrypt-0.9.0.jar, fontawesomefx-8.9.jar]. Please don't publish this project to a public artifact repository! *
[WARNING] *****
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 22 source files to C:\VUT\2. ročník ZS\BPC-BDS\3.projekt\project-demo2\BDSprojekt3\target\classes
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ project-demo ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\VUT\2. ročník ZS\BPC-BDS\3.projekt\project-demo2\BDSprojekt3\src\test\resources
```

Obr. 12 – mvn clean install

```
C:\Windows\System32\cmd.exe - java -jar project-demo-1.0-SNAPSHOT.jar
[INFO] -----
[INFO] Total time: 4.799 s
[INFO] Finished at: 2021-12-23T15:29:27+01:00
[INFO] -----

C:\VUT\2. ročník ZS\BPC-BDS\3.projekt\project-demo2\BDSprojekt3>cd target
C:\VUT\2. ročník ZS\BPC-BDS\3.projekt\project-demo2\BDSprojekt3\target>java -jar project-demo-1.0-SNAPSHOT.jar
LoginController initialized
```

Login to database







Car Factory Employees App

Employee mail:

Password:

Obr. 13 – spuštění přes java -jar

The database will contain the user passwords in a hash form (hashed using Argon2 or BCrypt).

| Data Output | | Explain | Messages | Notifications | | |
|-------------|--|--|---|--|--|--|
| |  employee_id [PK] bigint |  first_name character varying (45) |  surname character varying (45) |  mail character varying (45) |  pswd character varying (64) |  building_id [PK] bigint |
| 1 | 1 | Alena | Samková | alena.samkova@automotive.cz | \$2a\$10\$HsjHlIRhfh1.uopCQaARsOzlwsquffmzXxNkpuUb0kCbs9xkQ5IYW | 2 |
| 2 | 2 | Petr Erik | Kalivoda | petr.kal@automotive.cz | \$2a\$10\$HsjHlIRhfh1.uopCQaARsOzlwsquffmzXxNkpuUb0kCbs9xkQ5IYW | 3 |
| 3 | 3 | Alena | Jandová | alena.jand@automotive.cz | \$2a\$10\$HsjHlIRhfh1.uopCQaARsOzlwsquffmzXxNkpuUb0kCbs9xkQ5IYW | 4 |
| 4 | 4 | Alois | Smola | alois.smola@automotive.cz | \$2a\$10\$HsjHlIRhfh1.uopCQaARsOzlwsquffmzXxNkpuUb0kCbs9xkQ5IYW | 1 |
| 5 | 5 | Luboš | Veverka | lubos.vever@automotive.cz | \$2a\$10\$HsjHlIRhfh1.uopCQaARsOzlwsquffmzXxNkpuUb0kCbs9xkQ5IYW | 1 |
| 6 | 6 | Markéta | Janková | marketa.jank@automotive.cz | \$2a\$10\$HsjHlIRhfh1.uopCQaARsOzlwsquffmzXxNkpuUb0kCbs9xkQ5IYW | 5 |
| 7 | 7 | Jakub | Ferenc | jakub.ferenc@automotive.cz | \$2a\$10\$HsjHlIRhfh1.uopCQaARsOzlwsquffmzXxNkpuUb0kCbs9xkQ5IYW | 5 |
| 8 | 8 | Zbyněk | Hrnčíř | zbynek.hrncir@automotive.cz | \$2a\$10\$HsjHlIRhfh1.uopCQaARsOzlwsquffmzXxNkpuUb0kCbs9xkQ5IYW | 2 |
| 9 | 9 | Vojtěch | Sluka | vojtech.sluk@automotive.cz | \$2a\$10\$HsjHlIRhfh1.uopCQaARsOzlwsquffmzXxNkpuUb0kCbs9xkQ5IYW | 1 |
| 10 | 10 | Štěpán | Bečvár | stepan.becvar@automotive.cz | \$2a\$10\$HsjHlIRhfh1.uopCQaARsOzlwsquffmzXxNkpuUb0kCbs9xkQ5IYW | 4 |
| 11 | 11 | Šimon | Motl | simon.motl@automotive.cz | \$2a\$10\$HsjHlIRhfh1.uopCQaARsOzlwsquffmzXxNkpuUb0kCbs9xkQ5IYW | 1 |

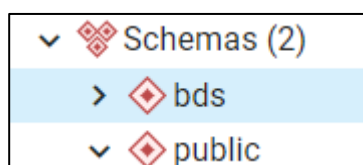
Obr. 14 – hesla ve formě hashů v BCrypt

The application will have a sign-in window with username/password authentication (if a user entered wrong credentials, the application will pop-up "Username or password is not valid"). This sign-in window won't be any fake login. It must be really validating the username + password from the database (consider loading also user roles – in enterprise systems, the roles during the authentication process are saved to the security context and used later for authorization). – viz kapitola 3

Create a database user role for the application (not superuser role!) that can sign in to the database and have suitable privileges. Furthermore, create a different than "public" schema for your database. – byla vytvořena role „db_user“, které byla grantována práva pro připojení a CRUD operace. Bylo vytvořeno schéma „bds“.

```
datasource.url=jdbc:postgresql://localhost:5432/car_factory_5
datasource.username=db_user
datasource.password=stroje
```

Obr. 15 – připojení přes uživatele db_user



Obr. 15,5 – schéma bds

| | rolname name | rolsuper boolean | rolinherit boolean | rolcreatorole boolean | rolcreatedb boolean | rolcanlogin boolean | rolreplication boolean |
|----|---------------------------|---------------------|-----------------------|--------------------------|------------------------|------------------------|---------------------------|
| 5 | pg_read_all_settings | false | true | false | false | false | false |
| 6 | pg_read_all_stats | false | true | false | false | false | false |
| 7 | pg_stat_scan_tables | false | true | false | false | false | false |
| 8 | pg_read_server_files | false | true | false | false | false | false |
| 9 | pg_write_server_files | false | true | false | false | false | false |
| 10 | pg_execute_server_program | false | true | false | false | false | false |
| 11 | pg_signal_backend | false | true | false | false | false | false |
| 12 | postgres | true | true | true | true | true | true |
| 13 | teacher | false | true | false | false | true | false |
| 14 | student | false | true | false | false | true | false |
| 15 | db_user | false | true | false | false | true | false |

Obr. 16 – uživatel db_user v pgAdmin

For at least one entity create CRUD operations (create, read, update, delete). Do not forget to have it in the GUI (applies for all of the operations). Include for one entity the findAll operation. – viz kapitola 3

For one entity provide a detailed view (you have to use JOIN here). – aplikováno v DetailedView

```
public EmployeeDetailView findEmployeeDetailedView(Long employee_id) {
    try (Connection connection = DataSourceConfig.getConnection();
        PreparedStatement preparedStatement = connection.prepareStatement(
            sql: "SELECT e.employee_id,e.first_name,e.surname,e.mail,b.building_name,j.job_type,a.city,a.street_number,a.
                " FROM employee e JOIN building b ON b.building_id = e.building_id" +
                " LEFT JOIN employee_has_address eha ON eha.employee_id = e.employee_id" +
                " LEFT JOIN address a ON a.address_id = eha.address_id" +
                " JOIN employee_has_contract ehc ON ehc.employee_id = e.employee_id" +
                " JOIN job j ON j.job_id = ehc.job_id" +
                " WHERE e.employee_id = ?;"
        )) {
```

Obr. 17 – metoda pro detail view zaměstnance

Implement one operation in GUI that invokes more than one query and run them all in one transaction. If something failed, rollback that transaction. – využito v edit view

```
public void editEmployee(EmployeeEditView employeeEditView) { //přidělat building
    String insertPersonSQL = "UPDATE employee e SET mail = ?, first_name = ?, surname = ? WHERE e.employee_id = ?";
    String checkIfExists = "SELECT mail FROM employee e WHERE e.employee_id = ?";
    try (Connection connection = DataSourceConfig.getConnection();
        // would be beneficial if I will return the created entity back
        PreparedStatement preparedStatement = connection.prepareStatement(insertPersonSQL, Statement.RETURN_GENERATED_KEYS)) {
        // set prepared statement variables
        preparedStatement.setString( parameterIndex: 1, employeeEditView.getMail());
        preparedStatement.setString( parameterIndex: 2, employeeEditView.getFirstName());
        preparedStatement.setString( parameterIndex: 3, employeeEditView.getSurname());
        preparedStatement.setLong( parameterIndex: 4, employeeEditView.getEmployee_id());
```

Obr. 18 – metoda pro edit view zaměstnance

Create a possibility to filter data about any selected entity (e.g., find persons by family name).
– zde jsem implementoval metodu, která mi ovšem shodila celý projekt. Metoda je v txt souboru na Githubu, bohužel však kvůli problémům s ní neimplementována v kódu.

Create a dummy table in the database and create a GUI window where you can simulate the SQL injection attacks. Simulate both (injection with drop table and injection where you retrieve more data than expected (e.g., 1=1)).

```
//SQL INJECTION  
employeeRepository.findByDataStatement("; DROP TABLE dummy_table --");
```

Obr. 19 – SQL Injection

Explain the necessity of PreparedStatements.

Nejen v Javě jde po napojení programu na SQL databázi z pravidla vykonávat SQL queries pomocí Statementů nebo Prepared Statementů. Statement je vhodné použít zejména v případě, kdy neočekáváme žádný input od uživatele, Prepared Statement pak v opačném případě, neboť zde jde nastavit string do SQL query. Statement je zranitelnější k útokům typu SQL injection a kód je při jeho využití hůře čitelný. Ve svém projektu využívám proto zejména Prepared Statement, neboť většinu queries vykonávám na základě nějakých uživatelských dat.

Create a script that will back up your database every midnight. – implementováno v souboru backup.txt na Githubu

Log exceptions suitably (e.g., use SLF4J Logback), set logs archiving per day (avoid log-andthrow antipattern). – využít SLF4J Logback.

```
SQL INJECTION  
PersonsController initialized
```

Obr. 20 – příklad logování

Create a GitLab/GitHub repository where your project will be placed. Do not forget to add .gitignore file that will ignore, e.g., the target folder and other unnecessary files. – Create a README.md file that will describe how the application can be built and run and what is the goal of that app. For example, inspire from <https://github.com/patrickfav/bcrypt> but you do not need to be so detailed. – Your GitLab/GitHub repository should have at least 5 commits

Repozitář projektu [zde](#).

Add a LICENSE file with a suitable license (e.g., consider MIT or Apache 2).

Viz repozitář na Githubu

Create/Or generate the document that lists all the external libraries that your project is including together with their license (e.g., use `mvn project-info-reports:dependencies`). For example check <https://gitlab.com/but-courses/bpc-bds/java-db-training/-/tree/master/licenses>. It is absolutely necessary when you or your company open-sourced or sell some software (check the licenses before you use any external library!).

Viz soubor `dependencies.html` v repozitáři.