Systemdokumentation Lunchify  
*Stand: Mai 2025*

Inhalt

[1. Architekturübersicht 2](#_Toc198828738)

[2. Datenbankmodell 2](#_Toc198828739)

[2.1. Tabelle: benutzer 2](#_Toc198828740)

[2.2. Tabelle: rechnung 2](#_Toc198828741)

[2.3. Tabelle: benutzer\_rechnung\_starred 3](#_Toc198828742)

[2.4. Tabelle: notification 3](#_Toc198828743)

[2.5. Tabelle: refund\_config 3](#_Toc198828744)

[3. Supabase Storage 3](#_Toc198828745)

[4. Sicherheitskonzepte 3](#_Toc198828746)

[4.1. RLS-Policies (Auszug) 4](#_Toc198828747)

[4.2. GUI-Komponenten & Styling 4](#_Toc198828748)

[4.3. Tests & Qualitätssicherung 4](#_Toc198828749)

[4.4. Prozesse & Abläufe 5](#_Toc198828750)

# Architekturübersicht

Lunchify besteht aus drei Schichten:

1. **Präsentation**
   * **JavaFX-Client** (Modul javafx.controls, javafx.fxml)
   * FXML-Views, Controller‐Klassen, CSS-Styles
2. **Geschäftslogik**
   * Controller- und Service‐Klassen (z. B. InvoiceDAO, UserDAO)
   * BCrypt für Passwort‐Hashing
3. **Datenhaltung**
   * **Supabase Postgres** mit Storage
   * Tabellen, Enums, RLS

Kommunikation: JDBC (DB) und HTTPS (Storage).

# Datenbankmodell

# Tabelle: benutzer

|  |  |  |
| --- | --- | --- |
| Spalte | Typ | Beschreibung |
| **id** | int8 PK | Eindeutige Benutzer-ID |
| **name** | text | Anzeigename des Users |
| **email** | text UQ | Login-E-Mail |
| **rolle** | text | Enum (admin / user) |
| **passwort** | text | BCrypt-Hash |
| **must\_change\_password** | bool | Flag: Passwort‐Änderung beim Login erforderlich |

# Tabelle: rechnung

|  |  |  |
| --- | --- | --- |
| Spalte | Typ | Beschreibung |
| id | int8 PK | Eindeutige Rechnungs-ID |
| user\_id | int8 FK → benutzer | Zugehöriger Benutzer |
| file\_url | text | Pfad/Dateiname im Supabase-Storage |
| type | text | Enum RESTAURANT / SUPERMARKET |
| invoice\_amount | numeric | Ursprünglicher Rechnungsbetrag (€) |
| reimbursement\_amount | numeric | Berechneter Rückerstattungsbetrag (€) |
| status | text | Enum SUBMITTED, APPROVED, REJECTED |
| upload\_date | date | Datum des Uploads |
|  |  |  |

# Tabelle: benutzer\_rechnung\_starred

|  |  |  |
| --- | --- | --- |
| Spalte | Typ | Beschreibung |
| **id** | int8 PK | Eindeutiger Eintrag |
| **benutzer\_id** | int8 FK → benutzer | User, der gefavorit hat |
| **rechnung\_id** | int8 FK → rechnung | Gefavoritete Rechnung |
| **starred\_date** | timestamp | Zeitstempel des Favorisierens |

# Tabelle: notification

|  |  |  |
| --- | --- | --- |
| Spalte | Typ | Beschreibung |
| **id** | int8 PK | Eindeutige Notifikation-ID |
| **user\_id** | int8 FK → benutzer | Empfänger |
| **message** | text | Benachrichtigungstext |
| **timestamp** | timestamp | Versand-/Erstellzeit |
| **erstatteter\_betrag** | numeric | Betrag der rückerstatteten Rechnung, falls zutreffend |

# Tabelle: refund\_config

|  |  |  |
| --- | --- | --- |
| Spalte | Typ | Beschreibung |
| **category** | text PK | Enum RESTAURANT / SUPERMARKET |
| **amount** | numeric | Max. Rückerstattung in € |

# Supabase Storage

* Bucket**: rechnung**
* Dateipfade**: <user>\_<timestamp>\_<originalname>.png/jpg/...**
* Öffentliche URL**:**
* https://<PROJECT>.supabase.co/storage/v1/object/public/rechnung//<trimmed-filename>
* RLS: Nur eigener Bucket; öffentliche Leserechte für Bilder zulässig.

# Sicherheitskonzepte

* Passwort: BCrypt hashing (BCrypt.hashpw, checkpw)
* Default-Passwort: default123 → sofort ändern
* RLS (Row-Level Security):
* Tabelle rechnung: policy select on rechnung for web using (user\_id = auth.uid());
* Tabelle benutzer\_rechnung\_starred: analog
* Roles: Postgres‐Enum benutzer\_rolle; Access via ?::benutzer\_rolle

# RLS-Policies (Auszug)

ALTER TABLE rechnung ENABLE ROW LEVEL SECURITY;

CREATE POLICY select\_own\_invoices

ON rechnung

FOR SELECT

USING (user\_id = auth.uid());

ALTER TABLE benutzer\_rechnung\_starred ENABLE ROW LEVEL SECURITY;

CREATE POLICY select\_own\_starred

ON benutzer\_rechnung\_starred

FOR SELECT

USING (benutzer\_id = auth.uid());

-- Admin-Rollen haben separate Policies (broad access)

# GUI-Komponenten & Styling

* FXML + eigene CSS-Klassen
* dashboard.css: Grid-Layouts, Farbvariablen (--primary, --accent)
* controlsfx: DatePicker-Skin, Tooltips

# Tests & Qualitätssicherung

* Unit Tests (JUnit5)
* InvoiceDAO (CRUD-Operationen gegen H2-InMemory)
* reimbursement = min(invoiceAmount, cap)
* UserDAO.validateLogin() (Mocked JDBC)
* Integrationstests
* HTTP-Upload + Download (mit lokalem Supabase-Emulator)
* UI-Smoke Tests
* TestFX: Login → Dashboard öffnen → Navigation
* Code Coverage
* Ziel ≥ 80 %

# Prozesse & Abläufe

* **Neuen User anlegen (Admin)**
* Insert in benutzer mit Flag must\_change\_password = TRUE
* **Erstlogin**
* validateLogin() liest must\_change\_password, zeigt Dialog
* updatePasswordAndClearFlag() setzt Flag auf FALSE
* **Rechnung hochladen (User)**
* Datei in Storage
* Insert in rechnung, Status=SUBMITTED
* **Favorisieren (User)**
* Toggle über DAO benutzer\_rechnung\_starred
* **Benachrichtigung**
* On status change: Insert in notification
* (Später) E-Mail-Versand