Systemdokumentation Lunchify

Team 3

**Stand:** Juni 2025  
**Version:** 1.2

Contents

[1 Architekturübersicht 3](#_Toc200974775)

[2 Datenbankmodell 4](#_Toc200974776)

[2.1 Tabelle benutzer 4](#_Toc200974777)

[2.2 Tabelle rechnung 4](#_Toc200974778)

[2.3 Tabelle benutzer\_rechnung\_starred 4](#_Toc200974779)

[2.4 Tabelle notification 5](#_Toc200974780)

[2.5 Tabelle notification\_preferences 5](#_Toc200974781)

[2.6 Tabelle refund\_config 5](#_Toc200974782)

[3 Supabase Storage 5](#_Toc200974783)

[4 Sicherheitskonzepte 6](#_Toc200974784)

[4.1 Passwort-Hashing 6](#_Toc200974785)

[4.2 Datenbank-RLS 6](#_Toc200974786)

[4.3 DB-Verbindung 6](#_Toc200974787)

[4.4 Benutzerrollen 6](#_Toc200974788)

[5 Applikationsschichten & Komponenten 6](#_Toc200974789)

[5.1 Präsentation (GUI) 6](#_Toc200974790)

[5.2 Geschäftslogik 7](#_Toc200974791)

[5.3 Datenhaltung 8](#_Toc200974792)

[6 Notification System 8](#_Toc200974793)

[6.1 Auslösung 8](#_Toc200974794)

[6.2 Anzeige 8](#_Toc200974795)

[7 Notification Preferences 8](#_Toc200974796)

[8 Tests & Qualitätssicherung 9](#_Toc200974797)

[9 Build & Deployment 9](#_Toc200974798)

[10 Prozesse & Abläufe 10](#_Toc200974799)

[10.1 User-Registration (Admin) 10](#_Toc200974800)

[10.2 Erstlogin 10](#_Toc200974801)

[10.3 Invoice Upload (User) 10](#_Toc200974802)

[10.4 Favorisieren 10](#_Toc200974803)

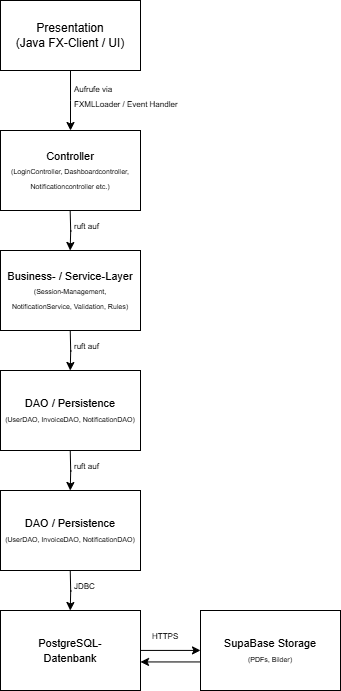
[10.5 Notification-Flow 10](#_Toc200974804)

[10.6 Notification Preferences 10](#_Toc200974805)

[10.7 Invoice Edit (Admin) 10](#_Toc200974806)

[11 Appendix: Enums 10](#_Toc200974807)

1. Architekturübersicht



Lunchify ist eine klassische **3-Schichten-Architektur**:

* **Präsentation**
  + JavaFX-Client (Module javafx.controls, javafx.fxml)
  + FXML-Views, Controller-Klassen, CSS-Styles
* **Geschäftslogik**
  + Controller- und DAO-Klassen (z. B. InvoiceDAO, UserDAO, NotificationDAO)
  + Session-Management (Session-Singleton)
  + Passwort-Hashing mit BCrypt
* **Datenhaltung**
  + Supabase PostgreSQL + Storage
  + JDBC-Kommunikation für DB, HTTPS für Storage
  + RLS-Policies auf Tabellenebene zur Zugriffskontrolle

1. Datenbankmodell
   1. Tabelle benutzer

|  |  |  |
| --- | --- | --- |
| Spalte | Typ | Beschreibung |
| id | bigint | PK: Benutzer-ID |
| name | text | Anzeigename |
| email | text | UQ: Login-E-Mail |
| rolle | text | Enum: admin / user |
| passwort | text | BCrypt-Hash |
| must\_change\_password | boolean | Flag: Passwortänderung erforderlich |

* 1. Tabelle rechnung

|  |  |  |
| --- | --- | --- |
| Spalte | Typ | Beschreibung |
| id | bigint | PK: Rechnungs-ID |
| user\_id | bigint | FK → benutzer(id): Zugehöriger User |
| file\_url | text | Pfad im Supabase-Bucket |
| type | text | Enum: RESTAURANT / SUPERMARKET |
| invoice\_amount | numeric | Ursprünglicher Betrag (€) |
| reimbursement\_amount | numeric | Berechneter Erstattungsbetrag (€) |
| status | text | **Enum**: SUBMITTED, REJECTED, EDITED |
| upload\_date | date | Datum des Uploads |

* 1. Tabelle benutzer\_rechnung\_starred

|  |  |  |
| --- | --- | --- |
| Spalte | Typ | Beschreibung |
| id | bigint | PK |
| benutzer\_id | bigint | FK → benutzer(id): wer gefavorit hat |
| rechnung\_id | bigint | FK → rechnung(id): welche Rechnung |
| starred\_date | timestamp | Zeitpunkt des Favorisierens |

* 1. Tabelle notification

|  |  |  |
| --- | --- | --- |
| Spalte | Typ | Beschreibung |
| id | bigint | PK |
| user\_id | bigint | FK → benutzer(id): Empfänger |
| message | text | Benachrichtigungstext (z. B. “Rechnung editiert”) |
| timestamp | timestamp | Erstellungs-/Versandzeit |
| erstatteter\_betrag | numeric | Optional: Betrag der bearbeiteten Rechnung |

* 1. Tabelle notification\_preferences

|  |  |  |
| --- | --- | --- |
| Spalte | Typ | Beschreibung |
| user\_id | bigint | FK → benutzer(id) |
| type | text | Enum: INVOICE\_EDITED, INVOICE\_REJECTED |
| enabled | boolean | Ein/Aus-Flag |

* 1. Tabelle refund\_config

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

1. Supabase Storage

* **Bucket**: rechnung
* **Dateinamensschema**:

<userId>\_<timestamp>\_<originalFilename>.<ext>

* **Öffentliche URL**:

https://<PROJECT>.supabase.co/storage/v1/object/public/rechnung/<trimmed-filename>

* **RLS**:
  + Jeder User darf nur seine eigenen Dateien hochladen
  + Öffentliche Leserechte (nur im Lesemodus) für Images

1. Sicherheitskonzepte
   1. Passwort-Hashing
   * BCrypt (org.mindrot.jbcrypt.BCrypt)
   * Default-Passwort default123 → Flag must\_change\_password = TRUE
   1. Datenbank-RLS
   * **rechnung**:

ALTER TABLE rechnung ENABLE ROW LEVEL SECURITY;

CREATE POLICY select\_own\_invoices

ON rechnung FOR SELECT

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

* + **benutzer\_rechnung\_starred**: analog
  1. DB-Verbindung
  + SSL-verbindung via JDBC-URL-Parameter.
  1. Benutzerrollen
  + Postgres-Enum benutzer\_rolle (admin, user).

1. Applikationsschichten & Komponenten
   1. Präsentation (GUI)

* **FXML-Views**
  + UserDashboardView.fxml
    - Linke Sidebar mit Logo (skaliert auf 80 px), Navigationstasten, Logout.
    - Oben-rechts: Notification-Button (orange, rund), “ADD NEW +”-Button, Username.
    - Zwei TilePane-Bereiche: Starred / Recently Viewed.
  + SettingsView.fxml
    - Sektionen: Account Settings (Change Password, Update Profile, Notification Preferences), App Settings (Language).
  + NotificationPreferencesView.fxml
    - Modal mit CheckBox-Toggles für jede Notification-Art.
  + NotificationsView.fxml
    - Modal mit ListView der letzten Notifications (Datum + Text).
  + Weitere: LoginView.fxml, AdminDashboardView.fxml, RequestHistoryView.fxml, AddInvoiceView.fxml, u. a.
* **Controller-Klassen**
  + UserDashboardController
    - Füllt TilePanes über InvoiceDAO.
    - Öffnet modale Dialoge für Upload, Settings, Notifications.
  + SettingsController
    - Öffnet Change-Password, Update-Profile, Notification-Preferences Modals.
  + NotificationPreferencesController
    - Lädt/speichert Präferenzen via UserDAO.updateNotificationPref(...).
    - Feedback durch Fade-Transition des Labels.
  + NotificationsController
    - Initialisiert ListView mit NotificationDAO.getNotificationsForUser(...).
  1. Geschäftslogik
* **DAO-Klassen**
  + InvoiceDAO
    - CRUD-Operationen (findAllInvoices, getAllInvoicesByUser, toggleStar, addInvoice, updateInvoice, deleteInvoice, updateInvoiceStatus, findInvoicesByMonth).
    - Status-Logik: String-Enums in DB, Upper-/Lowercase Konvertierungen
  + UserDAO
    - validateLogin, emailExists, addUser, updatePasswordAndClearFlag, getAllUsers, findEmailByBenutzerId
    - Methoden getNotificationPref, updateNotificationPref
  + NotificationDAO
    - getNotificationsForUser(int userId):

SELECT id, message, timestamp, erstatteter\_betrag

FROM notification

WHERE user\_id = ?

ORDER BY timestamp DESC;

* **Services**
  + Session-Management: Session.setCurrentUser(...)
  + Business-Rules: Erstattungsbetrag = min(invoice\_amount, refund\_config.amount)
  1. Datenhaltung
* **Kommunikation**
  + JDBC für alle DAOs.
  + HTTPS-Client für Datei-Upload/Download (Supabase) drehs.
* **Konfiguration**
  + DB-URL, Credentials in application.properties oder Umgebungsvariablen.

1. Notification System
   1. Auslösung
   * Bei Statuswechsel einer Rechnung (REJECTED oder nach Admin-Edit → EDITED)
   * DAO-Methode updateInvoiceStatus(...) bzw. updateInvoice(...) erzeugt Eintrag in notification.
   1. **Persistenz**

INSERT INTO notification(user\_id, message, timestamp, erstatteter\_betrag)

VALUES (?, ?, NOW(), ?);

* 1. Anzeige
  + Glocken Button im Dashboard
  + Modal (NotificationsView) zeigt ListView mit Datum + Text
  + “Close” schließt den Dialog

1. Notification Preferences

* **UI**:
  + Im Settings-Modal unter “Notification Preferences”
  + Drei Toggle-Switches (jetzt: INVOICE\_EDITED, INVOICE\_REJECTED)
* **Tabelle**: notification\_preferences (s. Abschnitt 2.5)
* **DAO**:
  + UserDAO.getNotificationPref(userId, type) → Boolean
  + UserDAO.updateNotificationPref(userId, type, enabled) → Boolean
* **Controller**:
  + Lädt Werte in initialize()
  + Speichert bei jedem Toggle und zeigt kurzes Feedback

1. Tests & Qualitätssicherung

* **Unit Tests (JUnit 5)**
  + InvoiceDAO CRUD gegen H2 In-Memory
  + Erstattungslogik (min(invoiceAmount, cap))
  + UserDAO.validateLogin (Mocked JDBC)
* **Integrationstests**
  + HTTP-Upload/Download mit lokalem Supabase-Emulator
* **UI-Smoke Tests**
  + TestFX: Login → Dashboard → Navigation → Modals öffnen
* **Code Coverage**: Ziel ≥ 80 %
* **Linting & Analyse**
  + SonarQube Reports, Checkstyle, PMD

1. Build & Deployment

* **Build Tool**: Maven
* **JDK**: Java 17
* **Abhängigkeiten**:
  + javafx-controls, javafx-fxml v21
  + postgresql JDBC-Driver
  + jbcrypt für Hashing
  + controlsfx
  + junit-jupiter für Tests
* **Deployment**
  + mvn clean package
  + Setze Umgebungsvariablen: DB\_URL, DB\_USER, DB\_PASS, SUPABASE\_URL, SUPABASE\_KEY
  + Starte via java -jar target/Lunchify.jar

1. Prozesse & Abläufe
   1. User-Registration (Admin)
   * UserDAO.addUser(...) → Default-Passwort + Flag
   1. Erstlogin
   * LoginController prüft must\_change\_password
   * Modaler Dialog ChangePasswordView
   * UserDAO.updatePasswordAndClearFlag(...)
   1. Invoice Upload (User)
   * AddInvoiceView + FileChooser → Supabase ↑
   * InvoiceDAO.addInvoice(...) → Status = SUBMITTED
   1. Favorisieren
   * Klick auf Karte → InvoiceDAO.toggleStar(...)
   1. Notification-Flow
   * Statuswechsel auf REJECTED oder EDITED → NotificationDAO.insert(...)
   1. Notification Preferences
   * Settings → Toggle → Persist in notification\_preferences
   1. Invoice Edit (Admin)
   * RequestHistory → Edit-Modal → InvoiceDAO.updateInvoice(...) → Status = EDITED
   * Notification für betroffenen User
2. Appendix: Enums

**-- Benutzer-Rolle**

CREATE TYPE benutzer\_rolle AS ENUM ('admin', 'user');

**-- Rechnung-Kategorie**

CREATE TYPE rechnung\_kategorie AS ENUM ('RESTAURANT', 'SUPERMARKET');

**-- Rechnung-Status**

CREATE TYPE rechnung\_status AS ENUM (

'SUBMITTED',

'REJECTED',

'EDITED'

);

**-- Notification-Typen (für Preferences)**

CREATE TYPE notification\_type AS ENUM (

'INVOICE\_EDITED',

'INVOICE\_REJECTED'

);