Wiener Genossenschaften

Description

Web-App für Desktop und Mobile Geräte, die Wohnungsangebote von diversen Genossenschaften sammelt, anzeigt und regelmäßig aktualisiert. Kernstück sind der Scraper um die Daten zu sammeln und eine Liste im Mitgliederbereich mit Sortier-/Filterfunktionen und Alarm/Deadlines. Die Liste soll von den Usern angepasst/individualisiert werden können.
Durch die Web-App sollen User

- den Überblick während der Wohnungssuche behalten
- wichtige Fristen nicht versäumen
- neue Angebote nicht übersehen
- eine zentrale Informationsstelle bieten

Funktionen

Startseite

- Login mit Passwort (Passwort vergessen = Luxus)
- Signup, User Ändern, Passwort ändern
- kleine Beschreibung zu WG

Userbereich:

1. ALLGEMEINE SUCHE

- Listenanzeige aller Projekte/Wohnungen (nur 1-2 Genossenschaften als Beginn)
- Kartenanzeige mit Standorten
- Suchfunktion/Filtern nach
    - Planungsprojekte/Sofort verfügbare Wohnungen
    - Miete/Kaufen
    - Region/Bezirk
    - m² Nutzfläche
    - Freifläche
    - Kosten
- Suchagent/Suchprofil anlegen (e-mail Benachrichtigung bei neuen Wohnungsangeboten)

2. MERKLISTE(Watchlist) der Favorites

- markierte Projekte werden angezeigt + zusätzliche Infos können ergänzt werden
- Filter- & Sortierfunktionen
    - Datum der Anmeldung
    - Genossenschaft
    - Bezugsfertig ab
    - Vormerkung gültig bis
    - Alarm
    - Notiz

Visuals

- React
- MUI
- Canva

Milestones

1. bis 22.12.2023:

- Projektbeschreibung
- grobe Roadmap
- Projekt Infrastruktur
- Ablaufdiagramme

2. bis 12.01.2024:

- MongoDb anlegen + Models erstellen (Mongoose)
- Datenmodellierung + Beziehung zwischen den Models erarbeiten
- React-Project einrichten (with Vite + MUI)
- Backend: alle Routen + Controller
- Routen mit Postman testen

3. bis 19.01.2024

- Router für public- und private Routes
- Statische Seiten(views) erstellen und den Routen zuordnen
- MUI-Customizing: eigenes Farbschema implementieren

- State Management Store mit 'Zustand' erstellen

4. bis 26.01.2024

- Scraper fertigstellen
- Kacheln mit echten Daten befüllen
- Login-Prozess (soweit wie möglich Authentication)

5. bis 2.2.2024

- Immobilienobj. als Favorites speichern
- Scraper für freie Wohnungen erweitern
- Re-Login aus Token realiseren

6. bis 14.2.2024 (Prüfung)

Questions

- zentrale Fehlerbehandlung?
- muss ich zusätzlich zu email = unique in der Datenbank noch prüfen ob die email schon existiert??

Roadmap

1. Projektbeschreibung
2. Technische Vorbereitungen/Projekt Infrastruktur

- Backend Server aufsetzen
- DB vorbereiten
- Entwicklungsumgebung einschließlich der Installation von MongoDB + Mongoose.
- Umgebungsvariablen für sensible Informationen wie Datenbank-URIs und geheime Schlüssel, um die Sicherheit zu erhöhen.

3. Models - Datenbankmodelle erstellen

- user, project, favorite
- Datenmodellierung: Beziehung zwischen den Modellen
- wie sind Collections untereinander verknüpft?

4. Backend-Entwicklung: Routen + Controller

Backend-Routen:

USERS:

- getAllUsers
- getOneUser
- user/signup
- user/login
- user/unlock
- user/delete
- user ändern
- passwort ändern

PROJECT:

- project (create a project with scraped data)
- project ändern
- projekt löschen

FAVORITES:

- Favorite anlegen
- Favorite ändern
- Favorite löschen
- einen bestimmten Favorite von einem User anzeigen
- alle Favorites von einem User anzeigen

5. Ablauf der Authentication im Detail:

AUTHENTICATION:

i. **Node.js-Projekt einrichten:**

- bcrypt für das Hashen von Passwörtern
- jsonwebtoken für JWTs

ii. **User Model:**

- E-mail Adresse
- gehashtes Passwort
- Vor + Nachname

iii. **Signup:**

- Route '/signup' erstellen

- Der Benutzer sendet Data an den Server.
- Überprüfung: Passwort sollte laut mindestens 8 Zeichen haben
- Überprüfe, ob die E-Mail-Adresse bereits in der Datenbank existiert.
- Validierungsmethoden für Benutzerdaten, um sicherzustellen, dass nur gültige Daten in die Datenbank eingefügt werden
- Wenn alles OK, hashe das Passwort und speichere die Benutzerdaten in der Datenbank

iv. **Login:**

- Der User sendet seine E-Mail und Passwort an den Server.
- Überprüfe, ob die E-Mail-Adresse in der Datenbank existiert.
- Wenn ja, vergleiche das gehashte Passwort mit dem eingegebenen Passwort.
- Wenn übereinstimmend, erstelle einen JWT und sende ihn als Antwort zurück an den Client.

v. **JWT:**

- jsonwebtoken-library, um einen JWT zu erstellen.
- payload: User-ID, exp: Ablaufzeit

vi. **Passwörter hashen:**

- bcrypt um das Passwort zu hashen, bevor es in der Datenbank gespeichert wird.

vii. **Passwörter vergleichen:**

- bcrypt.compare() zum Vergleichen des eingegebenen Passworts mit dem in der Datenbank gespeicherten gehashten Passwort.

viii. **Middleware für die Authentifizierung:**

- Middleware, die den JWT überprüft und den Benutzer authentifiziert. Diese Middleware wird für geschützte Routen verwendet.

ix. **Geschützte Routen:**

- Geschützte Routen definieren, die nur zugänglich sind, wenn ein gültiger JWT vorhanden ist.

x. **Logout:**

- JWT auf dem Client löschen.

6. Routen mit Postman testen
7. Frontend-Entwicklung:

- Entwicklungsumgebung mit React, Vite, MUI einrichten
- statische Seiten für Start, Login, erste Ergebnisliste
- Router für public- und private Routes
- Zustandsmanagement (mit 'Zustand')
- grundlegende Benutzerfunktionen
  - Login
  - Signup
  - User-Änderung
  - Passwortänderung

    - Logout
    - (Passwort vergessen)
- Empfehlung: KONTROLLIERTE Formulare

8. MUI Customizing:

- eigenes Farbschema implementieren
- Layout für public + private Bereich gestalten

9. Userbereich entwickeln:

- Angebote in Kacheln darstellen
- Merkliste
- Filter- und Sortierfunktionen für die Merkliste
- Kartenansicht mit Standorten
- Dialog/Overlay fürs Anzeigen eines Projects auf der Map
- 'Zur Website' Link öffnet neuen Tab
- Passwort-Bestätigung Inputfeld evtl erst mit State einblenden, wenn ein neues Passwort eingegeben wurde
- Pagination

10. Scraping-Modul entwickeln:

- Backend-Scraping-Modul entwickeln
- Speicherung der gescrapten Daten in MongoDB
- Scraper für freie Wohnungen erweitern
- dynamische Inhalte implementieren basierend auf gescrapten Daten
- Vor-scrapen + cache für 1h (nicht bei jedem re-load)
- es gibt Schnittstellen zb openImmo damit sich andere Portale die Daten holen können. Evtl verwenden das manche Genossenschaften. http://www.openimmo.de/
- geschützte Route am Server, die man auch von außen aufrufen kann
- Projekte eindeutig identifizieren, um Duplikate zu vermeiden
- Projekt muss irgendwie 100% identifizierbar sein damit es nicht öfters angelegt wird: eventuell lässt sich etwas wie ObjektNummer finden beim scrapen
- Häufigkeit der Aktualisierung
- Mechanismen für das Behandeln von Fehlern und Wiederholungsversuchen beim Scraping

11. UI-Optimierung:

- Logo entwickeln mit Canva
- UI responsive (mobile first)

- Verbessere die UX durch Verwendung von Icons, klarem Design, Farben
- Errorhandling: Anzeigen von Fehlermeldungen in UI

12. Erweiterte Funktionen:

- Re-Login aus Token realiseren
- Alarm- und Deadlines-Funktionen

13. Testphase
14. Dokumentation:

- Code, APIs und alle Konfigurationen
- Anleitung für Entwickler und Benutzer
- Funktionen die noch implementiert werden sollen beschreiben:
  - Role Management: Admin Role
  - unlock User nach Registrierung
  -

15. Rechtliches:

- Scraping/Zusammenarbeit mit Genossenschaften

12. Hosting

Contributing

Peter Pruzina (Wifi-Wien)

Working hours

- 21.12. 17:00-19:30
- 22.12. 11:00-12:30 + 16:20-17:40 + 18.00-19:00
- 27.12. 11:00-13:30 + 17:00-19:00
- 28.12. 09:00-14:00
- 29.12. 09:00-11:30
- 30.12. 16:30-19:00
- 31.12. 10:00-13:30
- 02.01. 10:30-13:30
- 03.01. 09:00-11:30 + 12:30-14:00
- 08.-17.1. ca 20 Arbeitsstunden

- 18.1-5.2. ca 35 Arbeitsstunden