

Projektdokumentation – pokeapp_projekt



Projektname: pokeapp_projekt

Module: M294 (Frontend) & M295 (Backend)

Autor: Patrick Gruber

Abgabedatum: 19.07.2025

2. Inhaltsverzeichnis

2. Inhaltsverzeichnis.....	2
3. Projektidee / Elevator Pitch.....	3
4. Anforderungskatalog.....	3
5. Klassendiagramm.....	3
6. Storyboard & Navigation.....	5
7. Screen-Mockups.....	6
8. REST-Schnittstellenbeschreibung.....	6
9. Testplan.....	10
Frontend (React, Unit-Tests).....	10
Backend (Spring Boot, Unit-Tests).....	10
10. Installationsanleitung.....	11
11. Hilfestellungen / Quellen.....	14

3. Projektidee / Elevator Pitch

Als 1996 zum ersten Mal in Europa die ersten neue Pokémon Spiele für die Gameboy Konsole herauskam und ich einer der ersten gewesen bin, der solch ein Spiel gekauft und bis heute noch leidenschaftlich spielt, bin ich immer wieder auf die Herausforderung gekommen das ich meine Pokémon suchen muss. Deswegen bin ich auf die Idee gekommen eine App zu entwickeln, welche es mir erleichtert meine Pokémon zu verfolgen.

Die pokeapp soll den Anwender ermöglichen seine Pokémon zu verwalten, wenn dieser sich ein Überblick schaffen will von diesen. Vor allem wäre es sehr hilfreich das man die gefangenen Pokémon erfasst, wenn man mehrere Spiele besitzt und zwischen denen tauscht.

In der Version 1.0 sollen zuerst die Pokémon der ersten Generation vorhanden sein und nach dem release sollen die ersten Patches durchgenommen werden. Sobald die App funktioniert, werden weitere Editionen hinzugefügt werden.

4. Anforderungskatalog

1. Übersicht der Pokémon der ersten Generation.

1.1 Pokémon sollen anwählbar sein und sich ein Overlay öffnen. Per Overlay soll der User ein Pokémon erfassen können, in dem dieser das Formular vervollständigt.

2. Übersicht der gefangenen Pokémon

2.1 In der Übersicht soll der User die möglichkeit haben, seine Pokémon zu aktualisieren. Zusätzlich soll der User auch die möglichkeit haben per Knopfdruck das Pokémon zu entwickeln, solange dies eine Entwicklung besitzt

3. Übersicht der Boxen mit der möglichkeit die Pokémon per drag&drop zu verschieben

5. Klassendiagramm



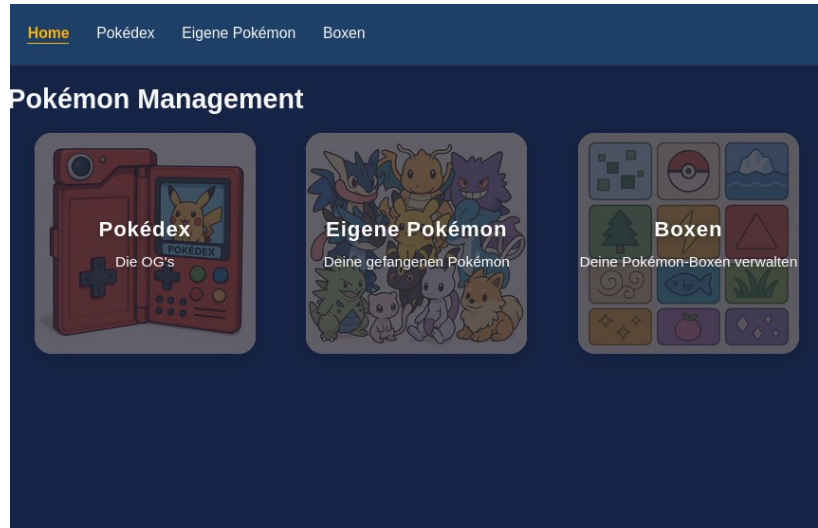
Das Klassendiagramm ist im Projektordner ersichtlich unter Dokumentation/Klassendiagramm

6. Storyboard & Navigation

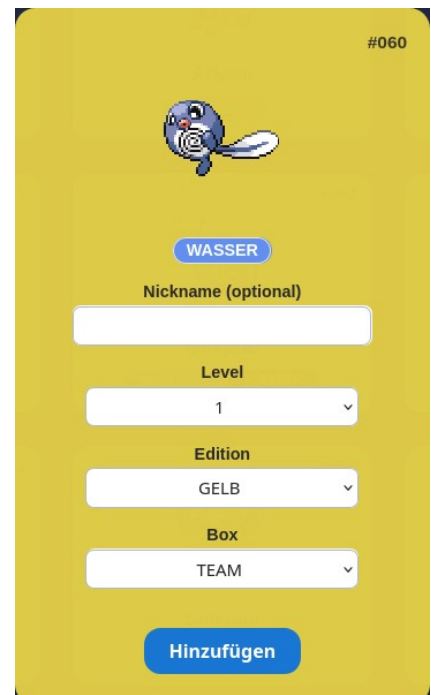
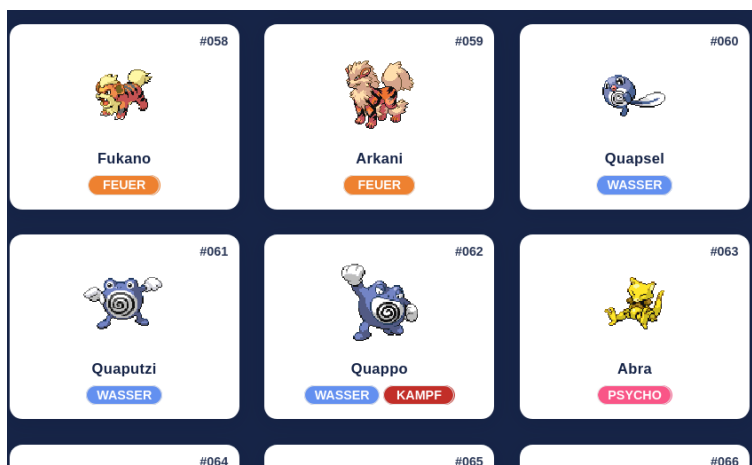
- *Die App startet auf der **Hauptseite**, wo der User über die Navigationsleiste zwischen den Hauptbereichen wählen kann:*
- **Startseite:** Übersicht mit Navigation zu Pokédex, Eigene Pokémon, Boxen.
- **Pokédex:** Alle Pokémon-Species werden als Grid-Ansicht mit Bild angezeigt.
 - Um ein bestimmtes Pokémon zu suchen, navigiert der User hierhin.
 - **Hinweis:** Aktuell gibt es nur die visuelle Suche per Bild. Eine Suchfunktion wird noch nachgerüstet.
 - Wenn das gewünschte Pokémon gefunden ist, klickt der User darauf.
 - Es öffnet sich ein **Overlay**, in dem die Edition und die Box ausgewählt werden müssen (Pflichtfelder).
 - Mit Klick auf „Hinzufügen“ wird das Pokémon zu den gefangenen Pokémon übernommen.
- **Eigene Pokémon:** Über die Navbar gelangt der User zu seinen eigenen, gefangenen Pokémon.
 - Hier werden alle gefangenen Pokémon aufgelistet.
 - Zum Ändern eines Pokémon klickt der User auf die entsprechende Karte, worauf ein Overlay erscheint, um Details wie Nickname, Level, Edition oder Box zu bearbeiten.
- **Boxen:** Über die Navbar kann die Boxen-Ansicht geöffnet werden.
 - Hier werden alle Pokémon nach Edition und Box sortiert angezeigt.
 - Auf der Seite kann der User oben über Dropdowns die gewünschte Edition und Box wählen.
 - Mit dem Button „Organisieren“ kann der User in den Drag&Drop-Modus wechseln und Pokémon zwischen den Boxen verschieben, um die Organisation zu erleichtern.

Beispielhafter Ablauf:

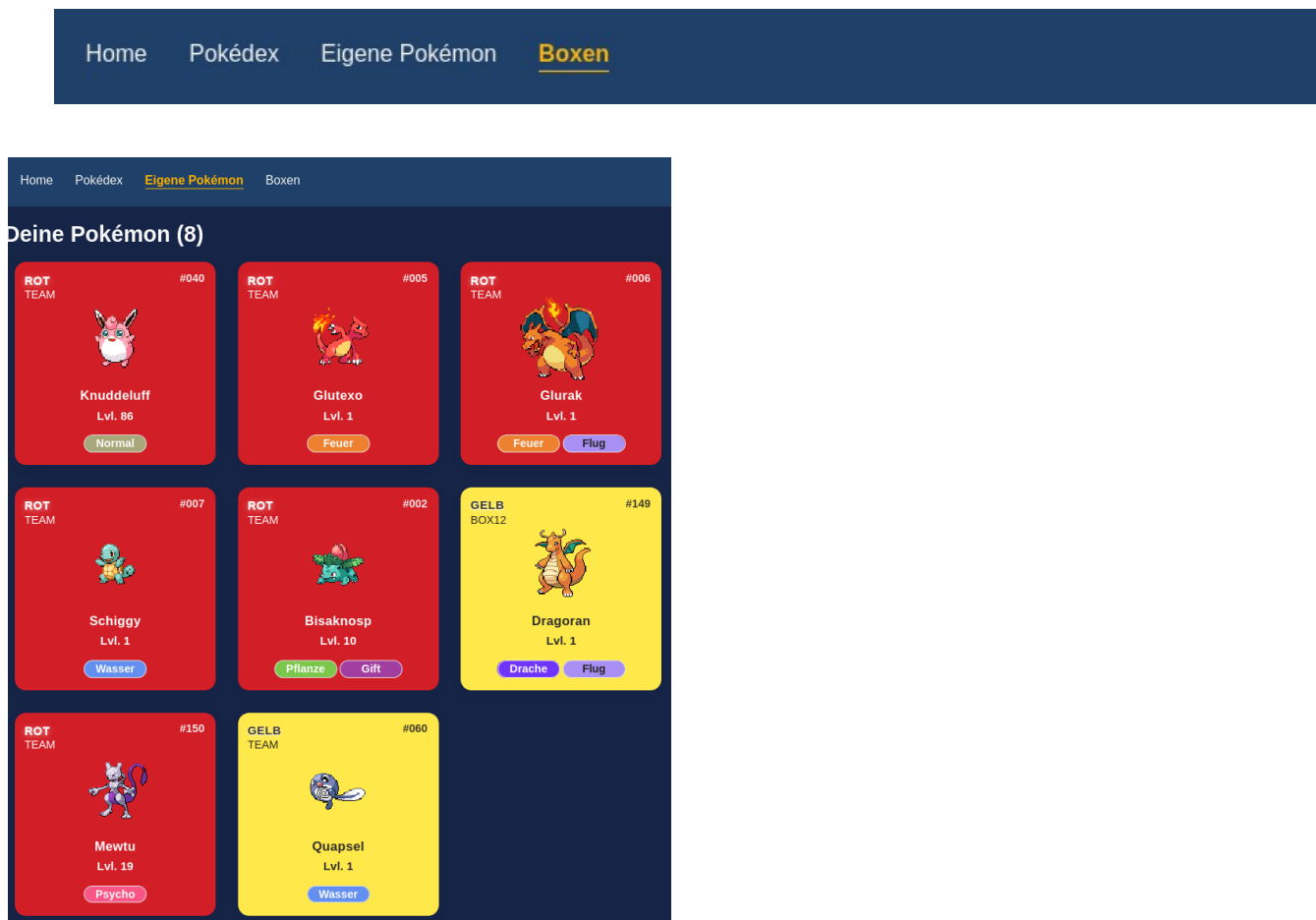
1. User startet auf der Hauptseite.



2. User navigiert zum Pokédex, sucht ein Pokémon, klickt darauf, wählt Edition und Box im Overlay, klickt „Hinzufügen“.



- Über die Navbar wechselt der User zu „Eigene Pokémon“ und sieht das gefangene Pokémon in der Übersicht.




- Zum Bearbeiten/Ändern erneut auf das Pokémon klicken, Details anpassen, speichern.



5. Um ein Pokémon zu verschieben, entweder auf der Box-Seite die Box im Overlay anpassen oder im Organisieren-Modus zwischen Boxen per Drag&Drop verschieben.

GELB TEAM

#060



Quapsel

Wasser

Lvl. 1

Aktualisieren

Löschen

Nickname

Level

1

▼

Edition

GELB

▼


Box

TEAM

▼

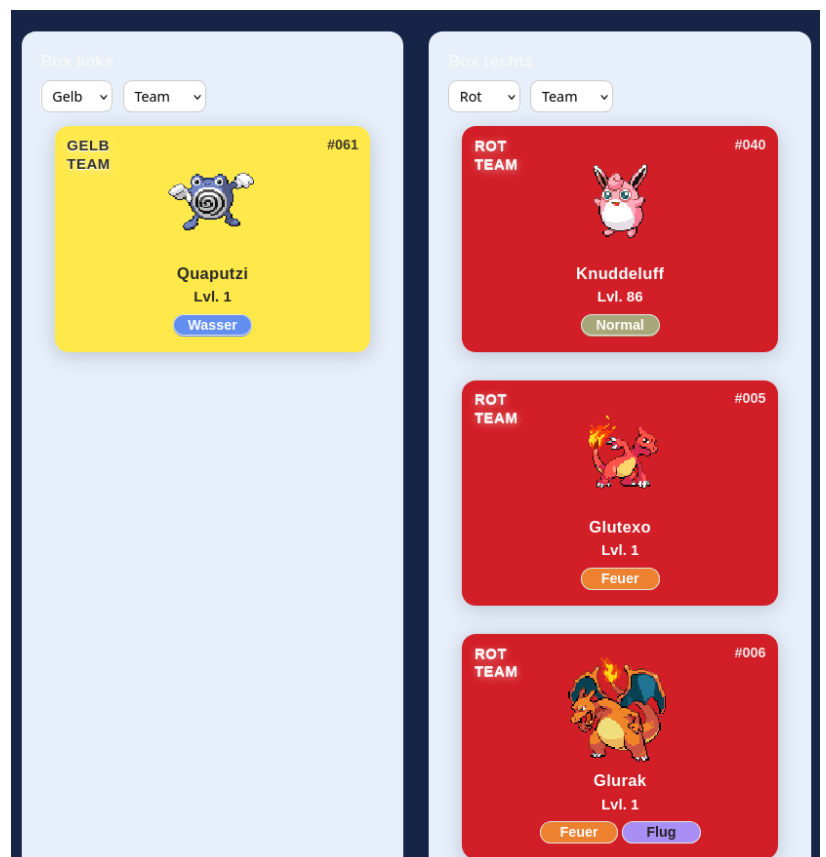
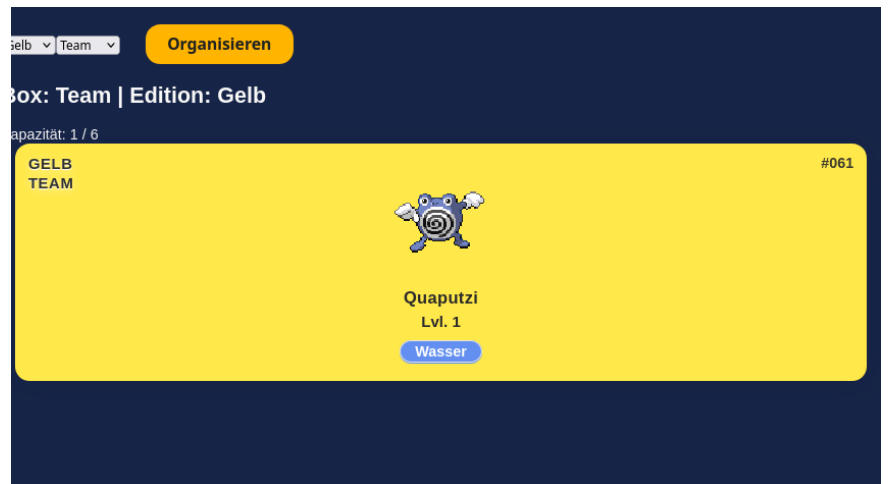
Speichern

Mögliche Entwicklungen

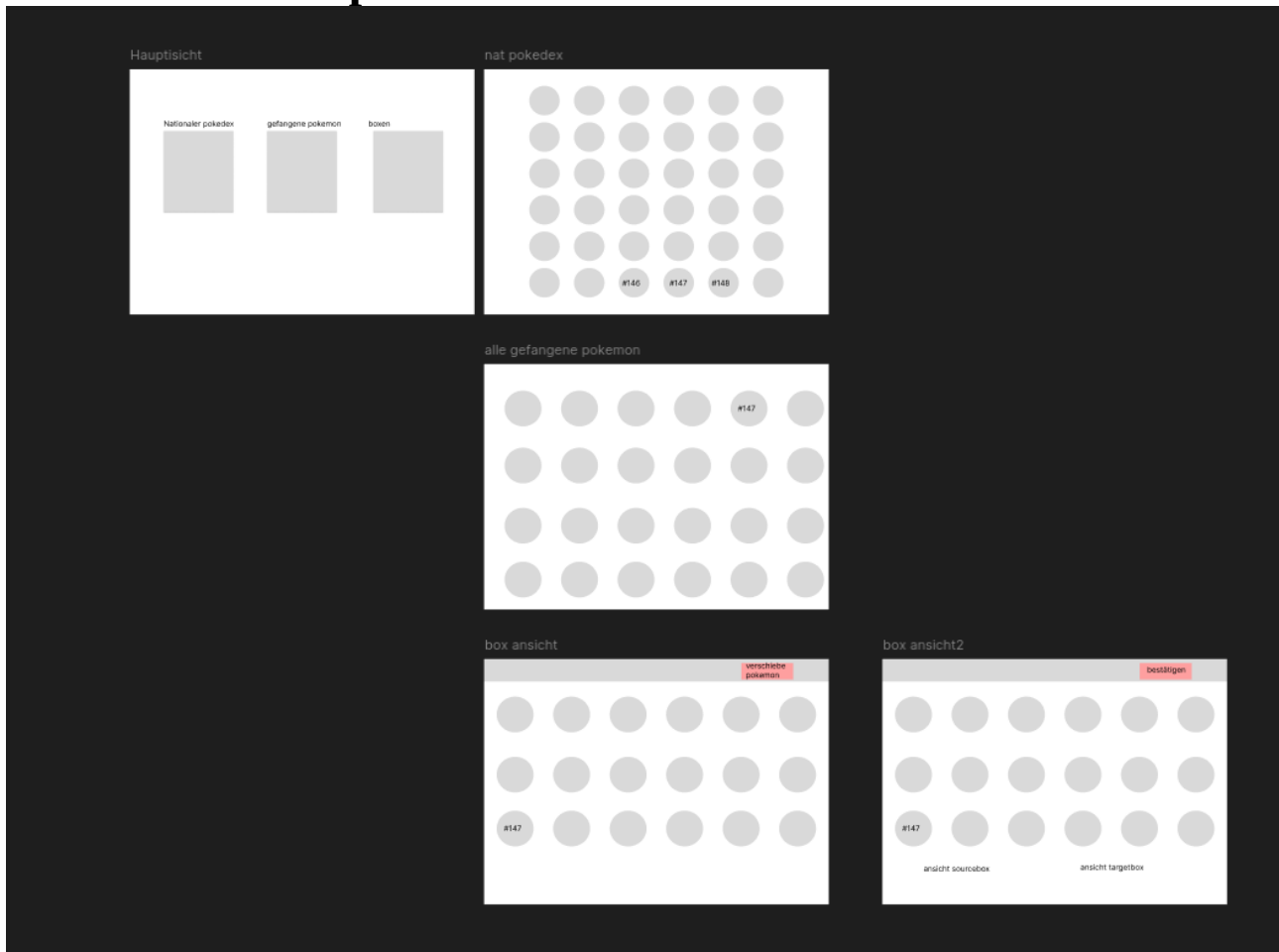


Quaputzi

Entwickeln



7. Screen-Mockups




Grobe Ideenskizze für das Frontend, um eine ungefähre Richtung zu haben.

8. REST-Schnittstellenbeschreibung

In den folgenden Bilder sind die Controller Klassen ersichtlich, sowie dessen Funktionen.

pokemon-species-controller			^
GET	/api/species	Gibt eine Liste aller 151 Pokémon zurück	▼
GET	/api/species/pokedex-id/{pokedexId}	Sucht eine Pokémon-Art anhand der Pokédex-ID	▼
GET	/api/species/name/{name}	Sucht Pokémon-Arten nach Name	▼
evolution-rules-controller			^
GET	/api/evolution-rules	Liefert alle Entwicklungsregeln der Pokémon	▼

owned-pokemon-controller			^
GET	/api/pokemon	Lädt alle gefangenen Pokémon	▼
POST	/api/pokemon	Fügt ein neues eigenes Pokémon hinzu	▼
GET	/api/pokemon/{id}	Holt ein gefangenes Pokémon anhand der ID	▼
DELETE	/api/pokemon/{id}	Entfernt ein gefangenes Pokémon aus dem Speicher	▼
PATCH	/api/pokemon/{id}	Aktualisiert ein gefangenes Pokémon	▼ 

enum-controller			^
GET	/api/editions	Listet alle verfügbaren Editionen auf	▼
GET	/api/editions/mapping	Liefert ein Mapping aus DisplayName zu Enum-Name für Editionen	▼
GET	/api/boxnames	Listet alle verfügbaren Boxnamen auf	▼
GET	/api/boxnames/mapping	Liefert ein Mapping aus DisplayName zu Enum-Name für Boxen	▼

box-controller



PUT

`/api/boxes/{sourceBox}/move-to/{targetBox}/{pokemonId}/
{sourceEdition}/{targetEdition}`

Verschiebt ein Pokémon von einer Box zu einer anderen (auch Editions übergreifend)



GET

`/api/boxes/{edition}/{name}` Lädt eine Box nach ihrem Namen und der Edition



GET

`/api/boxes/{edition}/{name}/is-full` Prüft, ob die Zielbox voll ist



GET

`/api/boxes/names` Gibt alle verfügbaren Box-Namen zurück



GET

`/api/boxes/editions` Gibt alle verfügbaren Editionsnamen zurück



Hier noch eine kurze zusammenfassung als Tabelle für eine bessere Übersicht

Endpoint	Methode	Beschreibung
/api/pokemon	GET	Gibt alle gefangenen Pokémon des Users zurück
/api/pokemon	POST	Fügt ein neues eigenes Pokémon hinzu
/api/pokemon/{id}	GET	Gibt ein bestimmtes gefangenes Pokémon zurück
/api/pokemon/{id}	PATCH	Bearbeitet ein eigenes Pokémon
/api/pokemon/{id}	DELETE	Löscht ein eigenes Pokémon
/api/species	GET	Gibt eine Liste aller verfügbaren Pokémon-Spezies zurück
/api/boxes/{edition}/{name}	GET	Gibt eine Box mit Pokémon für Edition und Name zurück
/api/boxes/{sourceBox}/move-to/{targetBox}/{pokemonId}/{sourceEdition}/{targetEdition}	PUT	Verschiebt ein Pokémon von einer Box zu einer anderen (auch zwischen Editionen)
/api/boxes/{edition}/{name}/is-Full	GET	Prüft, ob eine Box voll ist
/api/boxes/names	GET	Gibt alle verfügbaren Box-Namen zurück
/api/boxes/editions	GET	Gibt alle verfügbaren Editionen zurück
/api/evolution-rules	GET	Gibt alle Entwicklungsregeln der Pokémon zurück
/api/editions	GET	Listet alle verfügbaren Editionen auf
/api/editions/mapping	GET	Mapping von DisplayName zu Enum-Name für Editionen
/api/boxnames	GET	Listet alle verfügbaren Boxnamen auf
/api/boxnames/mapping	GET	Mapping von DisplayName zu Enum-Name für Boxen

Beispiel-Datenstruktur für ein Pokémon:

```
{
  "id": 0,
  "nickname": "string",
  "level": 0,
  "edition": "GELB",
  "boxName": "TEAM",
  "pokedexId": 0,
  "speciesName": "string",
  "type1": "string",
  "type2": "string"
}
```

9. Testplan

Frontend (React, Unit-Tests)

Nr.	Testfall-Beschreibung	Erwartetes Ergebnis	Ergebnis
1	HomePage: Drei große Navigationskarten werden angezeigt	„Pokédex“, „Eigene Pokémon“, „Boxen“ sind sichtbar	✓
2	HomePageClick: Klick auf „Pokédex“-Karte löst Navigation aus	Navigation auf Pokédex-Seite wird getriggert	✓
3	OrganizeView: Box-Auswahl für links und rechts wird angezeigt	Beide Seiten zeigen Edition/Box-Auswahl (je 2 Comboboxen)	✓
4	PokemonCard: Zeigt Nickname, Artname, Level und Typ	Alle Werte sind korrekt im DOM sichtbar	✓
5	PokemonCard (Snapshot): Card-Layout bleibt stabil	Snapshot-Test ist erfolgreich	✓
6	PokemonOverlay: Pflichtfelder im Add-Dialog werden angezeigt	Level, Edition, Box werden als Pflichtfelder angezeigt	✓
7	PokemonOverlayError: Fehlermeldung wird angezeigt, wenn Fehler gesetzt ist	Fehlermeldung wird korrekt dargestellt	✓

Backend (Spring Boot, Unit-Tests)

Nr.	Testfall-Beschreibung	Erwartetes Ergebnis	Ergebnis
1	OwnedPokemonController: Alle Pokémon abfragen	Liste von Pokémon-Objekten wird zurückgegeben	✓
2	OwnedPokemonController: Neues Pokémon speichern (korrekte Daten)	Pokémon wird erstellt und als JSON zurückgegeben	✓
3	OwnedPokemonController: Pokémon löschen	Pokémon wird erfolgreich gelöscht	✓
4	OwnedPokemonController: Löschen eines nicht existierenden Pokémon	Fehlermeldung mit passender Nachricht	✓
5	BoxService: Box abfragen (vorhanden/nicht vorhanden)	Richtige Box wird gefunden, sonst Fehlermeldung	✓
6	BoxService: Prüfen ob Team/Box voll oder nicht voll	Gibt korrekt true/false zurück	✓
7	BoxService: Pokémon in Box verschieben, Fehlerfälle abdecken	Exception/Erfolg je nach Bedingung	✓
8	BoxService: Erfolgreiches Verschieben in neue Box	Pokémon bekommt neue Box/Edition, wird gespeichert	✓
9	OwnedPokemonService: Pokémon erfolgreich hinzufügen	Pokémon wird gespeichert	✓
10	OwnedPokemonService: Hinzufügen bei voller Box	BoxFullException wird geworfen	✓
11	OwnedPokemonService: Hinzufügen bei nicht existierender Species	NotFoundException wird geworfen	✓
12	OwnedPokemonService: Nickname erfolgreich ändern	Nickname wird gespeichert	✓
13	OwnedPokemonService: Level nicht absenkbar	InvalidUpdateException wird geworfen	✓

10. Installationsanleitung

1. Voraussetzungen

- Code-Editor, zum Beispiel IntelliJ IDEA (empfohlen)
- Java JDK Version 21 (zwingend)
- Maven (aktuelle Version)
- Node.js Version 18 oder neuer
- npm (wird mit Node installiert)
- MariaDB:
 - Muss auf Port 3306 laufen
 - Benutzer: root
 - Passwort: root
 - Datenbank: pokeapp
 - Wichtig: Andere Werte sind möglich, müssen aber in der Datei application.properties im Backend angepasst werden!

2. Projekt entpacken und öffnen

1. Das ZIP-Archiv herunterladen und entpacken.
2. Im Code-Editor (zum Beispiel IntelliJ IDEA) den entpackten Projektordner „pokeapp_projekt“ öffnen.
Die Ordnerstruktur ist wie folgt:
 - backend
 - frontend

3. Datenbank (MariaDB) einrichten

1. MariaDB installieren und starten (Standardport: 3306).
2. Datenbank anlegen (CREATE DATABASE IF NOT EXISTS pokeapp;)
3. Hinweis: Andere Datenbanknamen oder Benutzer/Passwörter sind erlaubt, müssen aber in der Datei „backend/src/main/resources/application.properties“ entsprechend angepasst werden

4. Backend starten

1. Ein Terminal oder die Konsole öffnen und ins Backend-Verzeichnis wechseln:
`cd backend`
2. Backend mit Maven starten:
`mvn spring-boot:run`
3. Das Backend ist erreichbar unter:
<http://localhost:8080/swagger-ui/index.html#/>

5. Frontend starten

1. Ein neues Terminal oder eine neue Konsole öffnen und ins Frontend-Verzeichnis wechseln:
`cd frontend`
2. Abhängigkeiten installieren:
`npm install`
3. Frontend starten:
`npm run dev`
4. Das Frontend ist erreichbar unter:
<http://localhost:5173>

6. Hinweise und Troubleshooting

- Das Backend muss vor dem Frontend laufen, damit alle Daten korrekt geladen werden können.
- Bei abweichenden Ports, Datenbanknamen oder Zugangsdaten unbedingt die Datei „backend/src/main/resources/application.properties“ anpassen.
- Die Datenbanktabellen werden beim ersten Start des Backends automatisch erstellt.
- Die Swagger-UI im Backend listet und beschreibt alle verfügbaren API-Endpunkte.
- Bei Problemen mit der Datenbankverbindung prüfen, ob MariaDB läuft und die Zugangsdaten stimmen.
- Falls die Ports 3306, 8080 oder 5173 bereits belegt sind, entweder andere Ports wählen und in den Konfigurationen anpassen oder die blockierenden Prozesse beenden.

Fertig! Die Anwendung ist nun einsatzbereit. Das Frontend kann für die Verwaltung und Organisation der Pokémon genutzt werden, das Backend stellt die nötigen Daten bereit.

11. Hilfestellungen / Quellen

Während der Arbeit an meinem Projekt habe ich verschiedene Hilfsmittel und Quellen genutzt:

- **Teams-Modul 294 und 295:**
Ich habe die im Unterricht zur Verfügung gestellten Unterlagen, Beispiele und Diskussionsmöglichkeiten im Teams-Kanal genutzt.
- **Künstliche Intelligenz (z. B. ChatGPT):**
Für Verständnisfragen, Erklärungen zu Technologien, das Formulieren von Texten und die Erstellung von Testplänen habe ich AI-basierte Hilfen verwendet.
- **W3Schools:**
W3Schools.com diente mir als Nachschlagewerk für HTML, CSS, JavaScript und weitere Webtechnologien.
- **Offizielle Dokumentationen:**
Ich habe regelmäßig die offiziellen Webseiten und Dokumentationen von React, Spring Boot, Maven, MariaDB und Node.js genutzt.
- **YouTube & TikTok:**
Video-Tutorials und Kurzclips auf YouTube und TikTok habe ich genutzt, um mir Programmierkonzepte und Lösungen anschaulich erklären zu lassen.
- **Mitschüler:**
Ich habe mich mit Mitschülern ausgetauscht, um Ideen zu finden, Probleme zu lösen und Lösungsansätze zu diskutieren.
-

Ende der Projektdokumentation