Google NOSERYOUNG ergon

üK-223 Gruppe 6

Full Stack Applikation

Autoren:
Michel Mahadeva
Simon Deuber
Jara de Rooij

Inhaltsverzeichnis

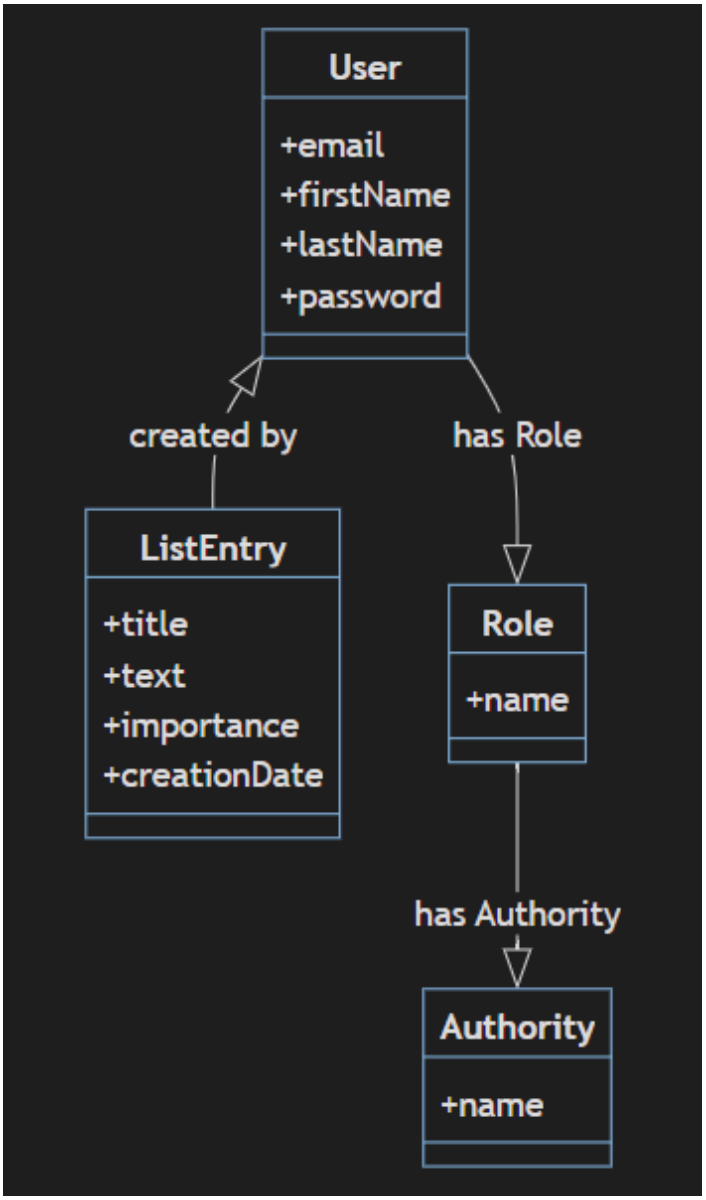
Kurzfassung

In dieser Dokumentation soll ein Teil der Planung und der Realisierung einer Multi-User-Applikation dargestellt werden.
Umfassend dabei sind gruppenspezifische Use-Cases für eine Liste, Beschreibung des Vorgehens beim Testing und einige Diagramme (ERD, Domain-Model, Use-Case-Diagram, Sequenz-Diagram).

1. Diagramme

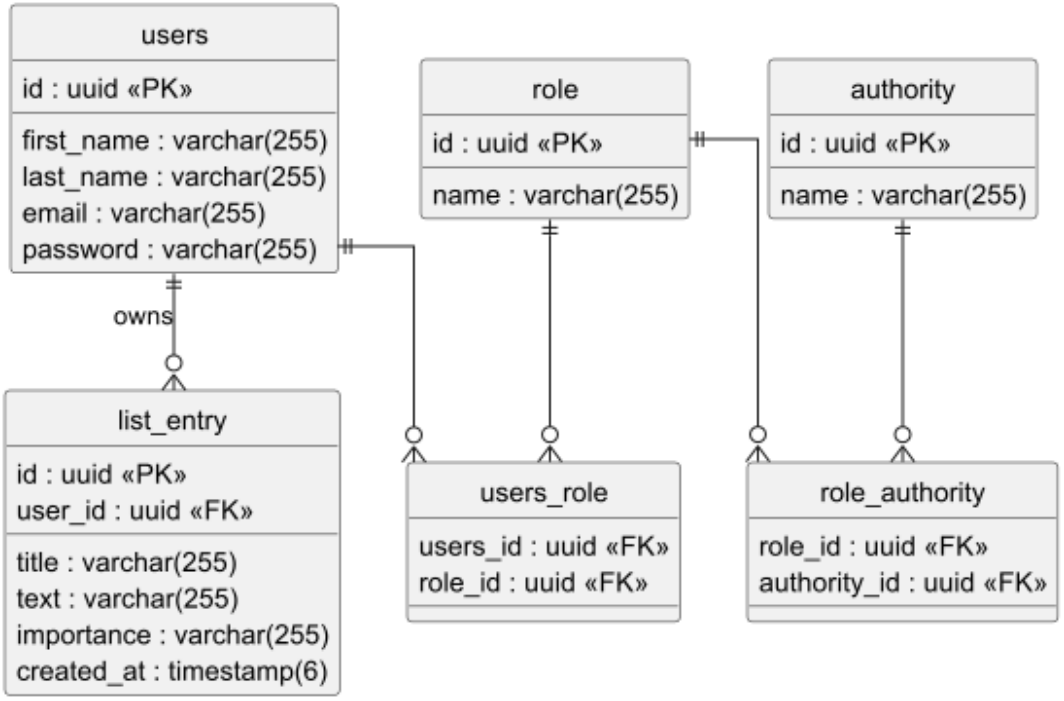
1.1. Domain-Modell

Wir haben ein Domain Model erstellt, das die zentralen Entitäten der Applikation beschreibt. Ein User kann Listeneinträge erstellen und besitzt Berechtigungen in Form von Authorities, welche über Role zugewiesen werden. Role enthalten dabei mehrere Authorities in einem und steuern, auf welche Funktionen ein User zugreifen darf. Das Domain Model dient als Grundlage für die restliche Struktur unseres Projekts



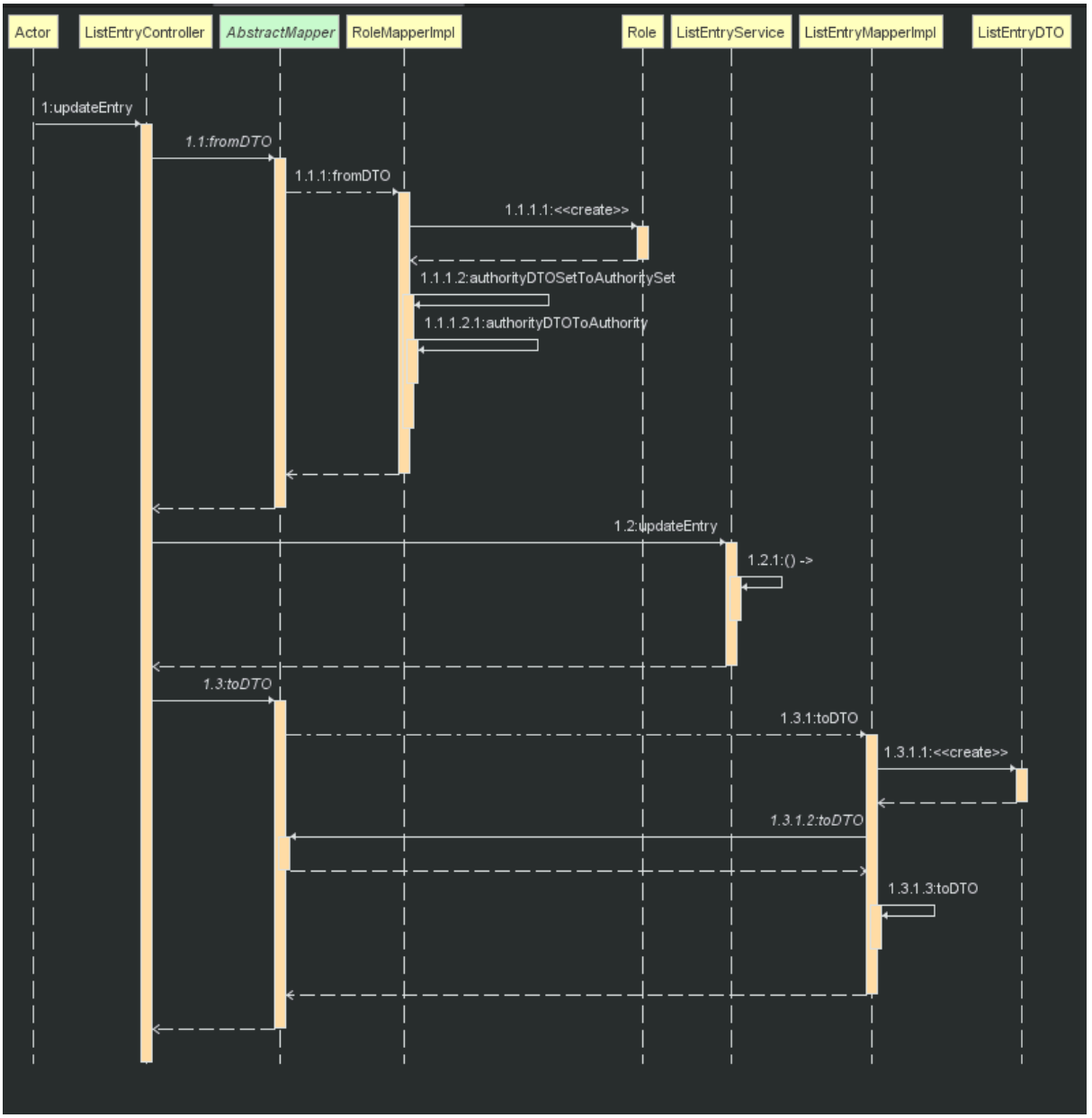
1.2. ERD

Wir haben ein ERD erstellt, das die Datenbankstruktur der Applikation abbildet. Es beinhaltet die Tabellen für Users, ListEntry, Role, Authority, users_role und role_authority zur Abbildung der n:m-Beziehungen. Jeder Listeneintrag gehört zu genau einem User, und die Rollen und Berechtigungen definieren über diese Beziehungen die Zugriffsrechte im System.



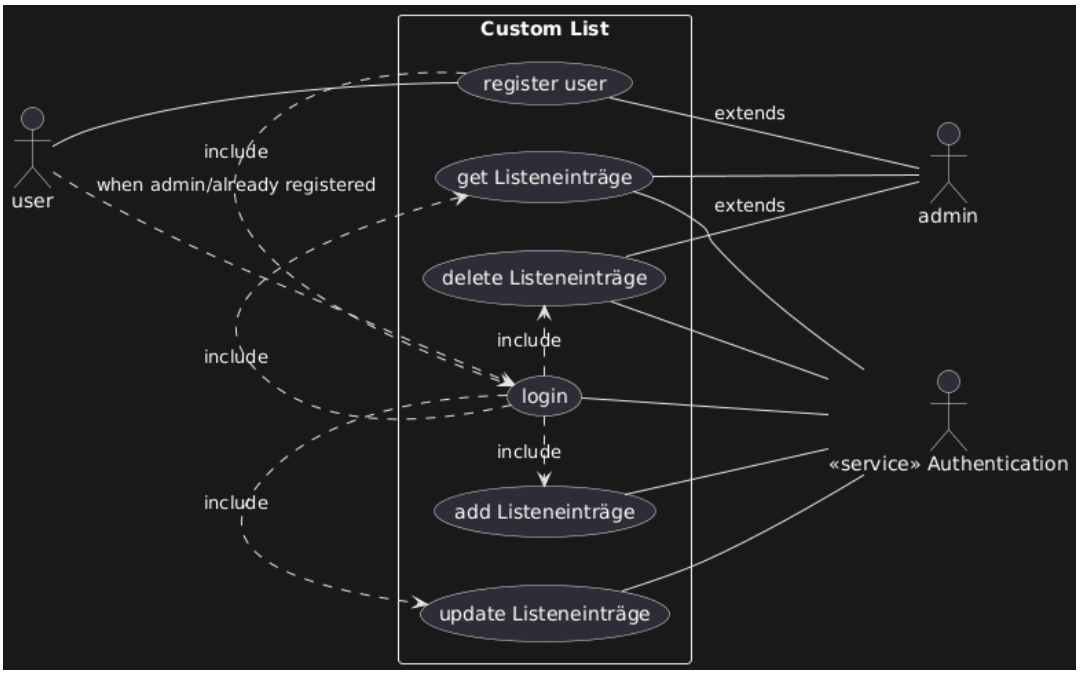
1.3. Sequence Diagram

Wir haben ein Sequenzdiagramm zum Aktualisieren eines Listeneintrags erstellt. Dabei wird der User über den JWT identifiziert und überprüft, ob er die Admin-Rolle hat, oder dem User der Listeneintrag gehört. Im Fall einer erfolgreichen Autorisierung zeigt das Diagramm den zeitlichen Ablauf. Bei fehlgeschlagener Autorisierung kriegt der User als Response: unauthorized.



1.4. UC-Diagram Aller Use Cases

Im Use-Case-Diagramm listen wir die zusätzlichen Funktionalitäten unserer Applikation auf und haben diese jeweils mit Actors verbunden.



2. Use Cases

2.1. Definition aller UCs

2.1.1. UC1

Actor	User
Description	User erstellt einen Listeneintrag
Preconditions	User hat sich eingeloggt und navigiert zur Seite mit seinen Einträgen
Postconditions	Entry wird persistiert (Erstelldatum wird automatisch gesetzt), Erfolgsmeldung erscheint, State im Frontend wird aktualisiert
Normal Course	User klickt auf add-Button, Formular mit Eingabefeldern Titel und Text sowie Dropdown Priority erscheint, User füllt Formular vollständig aus, Eingaben werden validiert (Titel mit mindestens 3 Zeichen, Text mit maximal 500 zeichen), User klickt erneut auf den add-Button, Daten werden gespeichert, User erhält toast mit Erfolgsmeldung
Alternative Course	Validierung schlägt fehl, add-Button wird deaktiviert, das/die betroffene/n Kästchen wird rot umrandet und ein kurzer Satz zum Grund erscheint (bspw. “title must be at least 3 characters”), Add-button wird erst bei erfolgreicher Validierung wieder aktiviert User entschliesst sich die Änderungen zu verwerfen und klickt auf den cancel-button, es werden keine Daten abgeändert
Exceptions	-

2.1.2 UC2

Actor	User
Description	User bearbeitet eigenen Listeneintrag
Preconditions	User hat sich eingeloggt und navigiert zur Seite mit seinen Einträgen
Postconditions	Verändertes Entry wird persistiert, State im Frontend wird aktualisiert
Normal Course	User klickt auf edit-Button, Formular mit Eingabefeldern Titel und Text sowie Dropdown Priority erscheint bereits ausgefüllt, User ändert Felder, Eingaben werden validiert (Titel mit mindestens 3 Zeichen, Text mit maximal 500 zeichen), User klickt erneut auf den save-Button, Daten werden aktualisiert, user erhält toast mit Erfolgsmeldung
Alternative Course	Validierung schlägt fehl, save-Button wird deaktiviert, das/die betroffene/n Kästchen wird rot umrandet und ein kurzer Satz

	zum Grund erscheint (bspw. “title must be at least 3 characters”), Save-button wird erst bei erfolgreicher Validierung wieder aktiviert User entschliesst sich die Änderungen zu verwerfen und klickt auf den cancel-button, es werden keine Daten abgeändert
Exceptions	Entry existiert nicht mehr: 404, NoSuchElementException Keine Berechtigung vorhanden (normaler User versucht, einen Eintrag eines anderen Users zu bearbeiten): 403 Unauthorized

2.1.3 UC3

Actor	User
Description	User löscht eigenen Listeneintrag
Preconditions	User hat sich eingeloggt und navigiert zur Seite mit seinen Einträgen
Postconditions	Eintrag wird in der DB gelöscht, State im Frontend wird aktualisiert
Normal Course	User klickt auf delete-Button, Bestätigungsdialog erscheint, User klickt auf confirm und erhält toast mit Erfolgsmeldung
Alternative Course	User entschliesst sich, den Eintrag doch nicht zu löschen, als der Bestätigungsdialog erscheint und klickt auf cancel, es werden keine Daten abgeändert
Exceptions	Entry existiert nicht mehr: 404

2.1.4 UC4

Actor	User
Description	User sieht eigene Listeneinträge sortiert und paginiert
Preconditions	User hat sich eingeloggt und navigiert zur Seite mit seinen Einträgen
Postconditions	Die 10 ersten Einträge werden je nach Sortierkriterium (Wichtigkeit, Erstelldatum, Titel) abgerufen, je nach Filter (Wichtigkeit) werden einige Einträge ausgelassen
Normal Course	Ein User klickt auf Sortier-Button und wählt ein Kriterium, die ersten 10 Einträge der sortierten Daten werden geladen, der User filtert nach einer Wichtigkeitsstufe (bspw. “High”) und nur Beiträge mit dieser werden angezeigt, User will mehr sehen und klickt auf das weitere-Icon, weitere 10 Datensätze werden geladen

Alternative Course	User will die Daten nicht sortiert haben, in dem Fall werden zu Beginn wie gewohnt die ersten 10 Datensätze geladen, danach kann der User mit Klick auf weiter-Button weitere 10 Datensätze unsortiert abrufen, Filtern ist weiterhin möglich
Exceptions	-

2.1.5 UC5

Actor	Admin
Description	Admin sieht Listeneinträge aller User
Preconditions	Admin hat sich eingeloggt, navigiert zur Listenseite und dann zur Adminseite
Postconditions	Die 10 ersten Einträge aller User werden je nach Sortierkriterium (Wichtigkeit, Erstelldatum, Titel) abgerufen, je nach Filter (Wichtigkeit) werden einige Beiträge ausgelassen
Normal Course	Admin klickt auf Sortier-Button und wählt ein Kriterium, die ersten 10 Einträge der sortierten Daten werden geladen, Admin filtert nach einer Wichtigkeitsstufe (bspw. “High”) und nur Beiträge mit dieser werden angezeigt, Admin will mehr sehen und klickt auf das weitere-Icon, weitere 10 Datensätze werden geladen
Alternative Course	Admin will einen beliebigen Eintrag löschen und klickt auf den delete-Button, ein Bestätigungsdialog wird geöffnet, Admin klickt auf confirm-Button und der Eintrag wird endgültig gelöscht, Admin erhält toast mit Erfolgsmeldung Admin will die Daten nicht sortiert haben, in dem Fall werden zu Beginn wie gewohnt die ersten 10 Datensätze geladen, danach kann der User mit Klick auf weiter-Button weitere 10 Datensätze unsortiert abrufen, Filtern ist weiterhin möglich
Exceptions	Entry existiert nicht mehr (beim Löschen): 404

3. Testing

3.1. Postman

Wir verwenden Postman für Integrations/Component Tests, da das Tool mithilfe des Collection Runners Abfolgen automatisiert testen kann.
**User: Jede Art von Rolle insofern nicht weiter beschrieben.*

Test Case	Use Case	Funktion/Endpoint	Rolle	Soll-Zustand
1.1	UC5 (Admin sieht Listeneinträge aller User)	GET /list-entries	Admin	Sieht sich alle Einträge an: 200
1.2			User	Sieht sich alle Einträge an: 403
2.1	UC2 (Um bestehende Daten anzusehen)	GET /list-entries/{id}	User	Schaut sich einen eigenen Eintrag an: 200
2.2			User	Schaut sich einen fremden Eintrag an: 403
2.3			Admin	Schaut sich einen eigenen Eintrag an: 200
2.4			Admin	Schaut sich einen fremden Eintrag an: 200
3.1	UC4 (User sieht eigene Einträge sortiert und paginiert)	GET /list-entries/user	User	Sieht sich seinen eigenen Eintrag an: 200
3.2			User	Sieht sich fremden Eintrag an: 403
3.3			Admin	Sieht sich fremden Eintrag an: 200
4.1	UC2 (User bearbeitet eigene Einträge)	PUT /list-entries/{id}	User	Bearbeitet eigenen Eintrag: 200
4.2			User	Bearbeitet fremden Eintrag: 403
4.3			Admin	Bearbeitet fremden Eintrag: 200
5.1	UC1 (User erstellt Eintrag)	POST /list-entries	User	Erstellt Eintrag: 201, created At wird automatisch gesetzt
6.1	UC3, UC5 (User löscht eigenen Eintrag, Admin kann Einträge aller User löschen)	DELETE /list-entries/{id}	User	Löscht eigenen Eintrag: 204
6.2			User	Löscht nicht-eigenen Eintrag: 403

3.2. Cypress

Wir verwenden Cypress für einen End-to-End-Test, in unserem Beispiel den kompletten Durchlauf von Login in verschiedenen Rollen, den Versuch, einen Eintrag zu erstellen bis hin zur Erfolgsmeldung nach erfolgreichem Hinzufügen.

3.2.1. Mögliche E2E Test-Cases nach Use Cases

Die folgende Tabelle enthält die von uns konzipierten Tests, die bei entsprechenden Zeitressourcen in das Projekt integriert worden wären. Aktuell haben wir uns auf die praktische Umsetzung von **UC1** konzentriert. Die weiteren Testfälle sind im aktuellen Projektumfang nicht enthalten, wurden jedoch zur Gewährleistung der **Vollständigkeit** bereits formal definiert.

Use Case	Funktion/Endpoint	Page	Rolle	Soll-Zustand
UC1 (User erstellt Eintrag)	POST /list-entries	/list/edit/list	User	User erstellt neuen Eintrag: 201, createdAt wird automatisch gesetzt und Erfolgsmeldung wird angezeigt
UC2 (User bearbeitet eigene Einträge um bestehende Daten anzusehen)	PUT /list-entries/{id} & GET /list-entries	/list/edit/list/{id} /list	User	User bearbeitet Eintrag und speichert diesen ab: 200. Danach sollte der User zur Listenübersicht weitergeleitet werden und die Änderung sollte bereits übernommen werden.
UC3 (User löscht eigenen Eintrag)	DELETE /list-entries/{id}	/list	User	User löscht eigenen Eintrag: 200, Seite wird neu geladen und gelöschter Eintrag wird nicht mehr angezeigt
UC4 (User sieht eigene Einträge sortiert und paginiert)	GET /list-entries/user	/list	User	User ruft /list auf und sieht eigene Einträge. Bei Wahl der Filter/Sortierung passt sich die Seite an
UC5 (Admin sieht Listeneinträge aller User, Admin kann Einträge aller User löschen)	GET /list-entries & DELETE /list-entries/{id}	/admin	Admin	Admin löscht fremden Eintrag: 200, Seite wird neu geladen und gelöschter Eintrag wird nicht mehr angezeigt
			User	User ruft page auf und wird weitergeleitet zu /unauthorized

3.2.2. Einzelne Test Cases zu UC1

Test Case	Use Case	Tool	What + Soll-Zustand
1	UC1 (User erstellt Eintrag)	Cypress	User erstellt neuen Eintrag: 201, createdAt wird automatisch gesetzt und Erfolgsmeldung wird angezeigt
2	UC1 (User erstellt Eintrag)	Cypress	User erstellt neuen Eintrag mit gültigem Input -> Frontend Validation blockiert den User vom Erstellen eines Listeneintrags. Error Messages werden dem User angezeigt um dessen Eintrag zu verbessern
3	UC1 (User erstellt Eintrag)	Cypress	Nicht angemeldeter User wird vom blockiert vom visit der Seite list/edit/list und wird zu /login umgeleitet.