

Exposé/ Projektarbeit/ Seminararbeit/ Bachelorarbeit

Internationale Hochschule Duales Studium

Studiengang: Informatik

**Generative Programmierung mittels künstlicher Intelligenz**

Müller Korbinian

Matrikelnummer: 102302316

Adresse

Lohwald Straße 59

86356 Neusäß

Betreuer/in (bei Bachelorarbeiten): XX

Abgabedatum: 12.03.

Inhalt

[Einleitung 4](#_Toc176359780)

[Theoretische Fundierung 4](#_Toc176359781)

[Begriffs Erklärung 4](#_Toc176359782)

[IDE 4](#_Toc176359783)

[Unit Test 4](#_Toc176359784)

[Merge Request 4](#_Toc176359785)

[Branch 4](#_Toc176359786)

[Methodik 4](#_Toc176359787)

[Fallstudie 4](#_Toc176359788)

[Voraussetzung 5](#_Toc176359789)

[Software Anforderungen 6](#_Toc176359790)

[Implementation 6](#_Toc176359791)

[Durch Entwickler umgesetzt 7](#_Toc176359792)

[Generative KI 8](#_Toc176359793)

[Analyse 8](#_Toc176359794)

[Implikationen 8](#_Toc176359795)

[Fazit 8](#_Toc176359796)

[Anhang 8](#_Toc176359797)

[A ) Dependencies "@emotion/styled": "^11.11.0", "@expo/vector-icons": "^13.0.0", "@expo/webpack-config": "^19.0.0", "@mui/icons-material": "^5.14.7", "@mui/material": "^5.14.7", "@mui/x-data-grid": "^6.12.1", "@mui/x-date-pickers": "^6.12.1", "@react-native-async-storage/async-storage": "1.18.2", "@react-native-community/netinfo": "^11.2.1", "@react-navigation/bottom-tabs": "^6.5.8", "@react-navigation/native-stack": "^6.9.13", "axios": "^1.5.0", "date-fns": "^2.30.0", "dayjs": "^1.11.9", "expo": "~49.0.8", "expo-status-bar": "~1.6.0", "moment": "^2.29.4", "react": "^18.2.0", "react-datepicker": "^4.16.0", "react-dom": "^18.2.0", "react-native": "^0.72.4", "react-native-gesture-handler": "~2.12.0", "react-native-safe-area-context": "4.6.3", "react-native-screens": "^3.27.0", "react-native-swipe-gestures": "^1.0.5", "react-native-swipe-list-view": "^3.2.9", "react-native-vector-icons": "^10.0.0", "react-native-web": "^0.19.8", "react-password-checklist": "^1.4.3", "react-swipe-to-reveal-actions": "^1.1.1", "react-swipeable": "^7.0.1", "react-swipeable-list": "^1.9.1", "react-table": "^7.8.0", "react-use": "^17.4.0", "zustand": "^4.4.1" 9](#_Toc176359798)

Einleitung

Theoretische Fundierung

Begriffs Erklärung

IDE

Die Integrated Development Environment, kurz IDE, ist eine der Handwerks mittel. Dabei handelt es sich um einen Texteditor für Programme. Dabei ist der Hauptaugenmerk bei dem Verwalten von Großen Software Projekten. Für die Technische Umsetzung wird die DIE PHP Storm von Jet Brains. Es wird die Professionelle Version verwendet.

Unit Test

Bei Unit Tests wird Software pragmatisch und sorgfältig getestet. Dabei werden insbesondere Klassen und Funktionen getestet. Die zu testenden Funktionen werden mit bekannten und Konstanten Werten aufgerufen. Die Ergebnisse werden mit den zu Erwartenden verglichen. Wenn diese Übereinstimmen gilt der Unit Test als erfolgreich. Erst wenn alle Test erfolgreich durchlaufen gilt der Unit Test als erledigt.

Merge Request

Bei einem Merge Request handelt es sich um ein Bestandteil im Arbeitsablauf. Dabei wird im Git einen neuen Branch angelegt. Auf dem Neuen Branch werden nun die Änderungen vorgenommen. Sobald alles Umgesetzt ist wird der Branch auf den Git Server geladen. Nun wird ein Merge Request gestellt. Dadurch wird beantragt die Änderungen in den Hauptstand zu übernehmen. Dabei werden die Änderungen von einem Kollegen überprüft. Sobald alles in Ordnung ist wird der Merge Request akzeptiert und die Änderungen gehen in die Produktion

Branch

Methodik

Fallstudie

Im Folgenden wird die Fallstudie zunächst einmal genauer beschrieben.

Voraussetzung

Die Grundlage für die Fallstudie ist ein Kundenprojekt bei dem Praxispartner. Der Kunde ist ein großer Obstbauernverband in Deutschland. Dieser beauftragte eine digitale Mitgliederverwaltung. Die Kunden, Bauern, werden hiermit erfasst. Neben der Anschrift, Firmenname und Kontaktdaten werden auch die Flächen registriert. Zu den Flächen wird dokumentiert wo sich diese Befinden und welche Obstsorte angebaut wird. Anhand dieser Information kann der Bauer Dienstleistungen buchen. Der Beitritt zu dem Verband ist für die Bauern kostenpflichtig. Dazu kommt noch eine Monatliche Gebühr. Eine der Dienstleistungen ist die Beratung. Dabei wird ein Termin ausgemacht und ein Beraten kommt vorbei. Diese Beratungen können individuell basierend auf den Früchten oder Flächen geschehen. Um den Beratern die Arbeit zu erleichtern wurde beschlossen eine Handy App für diese zu Erstellen. Diese Handy App ist die Grundlage für die Fallstudie.

Die Handy-App basiert auf dem Java-Script Framework React Native. Die Entscheidung ist auf dieses Gefallen da es ein sehr Modernes Framework ist. Ein weiterer Grund ist die Entwicklungsgeschwindigkeit. Das Projekt wird zu Nativen Apps für Android und IOS kompiliert und ist als solche für die Berater zum Download bereitgestellt.   
In der App werden die relevanten Daten nicht gespeichert. Dies Passiert alles Zentral in der Mitgliederverwaltung. Dazu wurde die Mitgliederverwaltung um einen Rest-Service erweitert. Die Kommunikation zwischen der App und dem Backend, geschieht über verschlüsselte HTTPS Anfragen, mit dem Package Axios.

Die App ist einfach gehalten. Der Berater loggt sich ein und kommt zum Dashboard. Hier findet die Hauptnavigation statt. Von hier aus gelangt er zu seinem Profil, einer Übersicht über alle verfügbaren Beratungstätigkeiten und der Maske zum Erfassen und Dokumentieren seiner Tätigkeit. Hier wird erfasst bei welchem Bauer, zu welcher Uhrzeit er war. Für die Tätigkeit selber wird Dokumentiert was er gemacht hat und wie lange es gedauert hat.

Einer der Wichtigsten Bestandteile der App ist die Stundenübersicht. Der Name der Seite ist ConsultingSessionList. Hier sieht der Berater welche Tätigkeiten er geleistet hat. Diese sind nach dem Datum, absteigen sortiert. Es gibt hier die Möglichkeit die einzelnen Elemente zu Sortieren nach Beratungstyp, Fruchttyp oder nach Bauer. Diese nach Bauer sortiere Ansicht ist die Grundlage für die Fallstudie. Hier soll am Oberen Bildschirmrand eine Kachel hinzugefügt werden. Die Kachel wird benutzt um die Daten des Bauern darzustellen. Dazu zählen: Name, Anschrift, Alle Flächen und Alle Früchte die Angebaut werden. Ebenso sollen zwei Butten hinzugefügt werden, die die Telefonnummer und die E-Mail Adresse anzeigen. Wenn auf die Button gepresst wird soll sich das Entsprechende Programm öffnen und die Daten übernehmen. Für die Umsetzung sollen Elemente der Material UI genommen werden

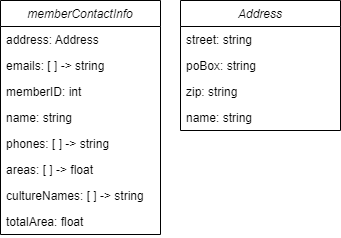
Der Name dieser Komponente ist ConsultingSessionMemberContactInfo. Diese Komponente wird in der ConsultingSessionList genutzt. Die ConsultingSessionList kümmert sich um das konditionelle Anzeigen der Komponente. Die entsprechenden daten werden ebenfalls in der List vom Serverabgerufen. Die Daten werden in der Form eines Prop mit dem Namen memberContactInfo übergeben  


Abbildung 1: Klassendiagramme für memberContactInfo

Software Anforderungen

Anhand dieser Projekt Konstellation gehen folgende Anforderungen an die App hervor.

* Die App soll Native auf Mobilgeräten laufen, hier kommen die Betriebssysteme Android und vor
* Als Programmiersprache wird Javascript in Verbindung mit dem Framework React Native eingesetzt
* Die Kommunikation findet über das Internet mittels HTTPS Request statt.
* Die Beratungsstunden App interagiert mit der Rest Service der Mitgliederverwaltung

Neben den Allgemeinen gibt es noch weitere Anforderungen in Bezug zu den verwendeten Software Bibliotheken. In diesem Projekt werden Folgende Node Module eingesetzt:

|  |  |
| --- | --- |
| Node | V20.16.0 |
| Dependencies | react:^18.2.0, react-native: ^0.72.4, expo: ~49.0.8 |
| Material UI | mui/material:^5.12.7, mui/icons-material: ^5.14 |

Implementation

Nun ist der Inhalt und der Aufbau des Projektes dargelegt wurden und die Anforderungen geklärt. Mit diesem Wissen geht es nun an die Implementation der UI-Komponente ConsultingSessionMemberContactInfo. Die Komponente wird zweimal Implementiert. Das erste Mal geschieht es rein durch den Entwickler. Das zweite mal wird ein Prompt and die KI gestellt und Iteriert bis ein funktionierendes Ergebnis erreicht wird.   
Bei beiden Entwicklungsansätze ist das Ziel gleich, eine Funktionierende Komponente. Die Komponente gilt als funktionierend, wenn diese Keine Fehler Produziert bei fehlerhaften Daten und Optisch genau so aussieht wie es im Screendesign vorgesehen ist. Das Sekundäre Ziel ist es herauszufinden, ob die generative AI Code in ausreichender Qualität erzeugt, sodass der Code in Produktion gehen kann. Das vor allem Schneller als der Entwickler.  
Der Ausgangspunkt ist gleich bei beiden Implementationen. In der Komponente ConsultingSessionList werden die Daten geladen und sind bereit. Die Komponente ist bereits inkludiert und an der Richtigen Stelle. Die Daten werden Ordnungsgemäß übergeben. Die Datei mit dem Namen ConsultingSessionMemberContactInfo.jsx ist angelegt. Ebenso wird ein eigener Branch für beide Implementationen angelegt.

Durch Entwickler umgesetzt

Die Umsetzung durch den Entwickler beginnt in der IDE PHP-Storm. Die Datei ist hier im Editor geöffnet. Zunächst wird das Übergebene Prop dekonstruiert. Dadurch sind alle Informationen separat in Variablen zugänglich. Anschließend folgt die HTML-Strukturierung. Das UI-Element wird in vier Bereiche Unterteilt. In der ersten sind Name und Mitglieds ID. Der Zweite Bereich enthält die Adressdaten. Alle Daten bezüglich der Flächen stehen im dritten Abschnitt. Im vierten sind die Buttons mit den Kontakt Möglichkeiten. Anhand dieser Aufteilung wird zunächst das MUI-Element Box verwendet, dieses dient als Container. Für jeden Abschnitt wird ein View-Element eingefügt. Damit ist die erste Strukturierung abgeschlossen.   
Der erste Bereich, der Programmiert wird ist der erste Container. Um das Styling zu vereinfachen wird das Output Text-Element in einem weitern View-Element verschachtelt. Für die Ausgabe des Namens und der ID wird ein Java-Template String verwendet. Dieser ist so Aufgebaut ´ ${memberContactInfo.name} [${memberContactInfo.memberID}`. Danach geht es an den zweiten Bereich. Hier können die Daten null sein, dies passiert wenn diese nicht in der Mitgliederverwaltung eingepflegt sind. Um einen Kritische Fehler zu vermeiden, der zu einem Absturz führt, muss überprüft werden ob alle Daten vorhanden sind. Dies geschieht mit der Abfrage !***Object***.values(address).every(value => value === '' || value === null). Falls diese den Boolische Wert Wahr annimmt wird der Inhalt ausgegeben. Zunächst einmal wird das Location Icon von MUI, <PlaceIcon>, ausgegeben. Im Anschluss die Adressdaten. Hier ist es zu beachten, dass es vorkommen kann

Generative KI

Analyse

Implikationen

Fazit

Anhang

A ) Dependencies

"@emotion/styled": "^11.11.0",  
"@expo/vector-icons": "^13.0.0",  
"@expo/webpack-config": "^19.0.0",  
"@mui/icons-material": "^5.14.7",  
"@mui/material": "^5.14.7",  
"@mui/x-data-grid": "^6.12.1",  
"@mui/x-date-pickers": "^6.12.1",  
"@react-native-async-storage/async-storage": "1.18.2",  
"@react-native-community/netinfo": "^11.2.1",  
"@react-navigation/bottom-tabs": "^6.5.8",  
"@react-navigation/native-stack": "^6.9.13",  
"axios": "^1.5.0",  
"date-fns": "^2.30.0",  
"dayjs": "^1.11.9",  
"expo": "~49.0.8",  
"expo-status-bar": "~1.6.0",  
"moment": "^2.29.4",  
"react": "^18.2.0",  
"react-datepicker": "^4.16.0",  
"react-dom": "^18.2.0",  
"react-native": "^0.72.4",  
"react-native-gesture-handler": "~2.12.0",  
"react-native-safe-area-context": "4.6.3",  
"react-native-screens": "^3.27.0",  
"react-native-swipe-gestures": "^1.0.5",  
"react-native-swipe-list-view": "^3.2.9",  
"react-native-vector-icons": "^10.0.0",  
"react-native-web": "^0.19.8",  
"react-password-checklist": "^1.4.3",  
"react-swipe-to-reveal-actions": "^1.1.1",  
"react-swipeable": "^7.0.1",  
"react-swipeable-list": "^1.9.1",  
"react-table": "^7.8.0",  
"react-use": "^17.4.0",  
"zustand": "^4.4.1"