LB 335: Mobile Applikationen realisieren

Aktien Portfolio

Elisa Petit-Sirigu

bbbaden

[Konzept 2](#_Toc153174184)

[Anforderungen 2](#_Toc153174185)

[Use Case-Diagramm 3](#_Toc153174186)

[Storyboard 3](#_Toc153174187)

[Mockup 3](#_Toc153174188)

[Usability-Konzepte 3](#_Toc153174189)

[Vorgehen 3](#_Toc153174190)

[Systemdokumentation 4](#_Toc153174191)

[Anhang 5](#_Toc153174192)

[Quellen 5](#_Toc153174193)

## Konzept

### Anforderungen

|  |  |  |  |
| --- | --- | --- | --- |
| Nummer | Muss / Kann | Funktional? Qualität? Rand? | Beschreibung |
|  | Muss | Funktional | Über eine API soll meine App Zugriff auf aktuelle Börsendaten haben. |
|  | Muss | Funktional | Die App soll vier Screens haben. |
|  | Muss | Funktional | Eine Navigation soll vorhanden sein, über die der Benutzer auf die unterschiedlichen Seiten wechseln kann. |
|  | Muss | Funktional | Der erste Screen ist für die erstellte Liste. |
|  | Muss | Funktional | Der zweite Screen ist für die Suche nach Anlageoptionen |
|  | Muss | Funktional | Der dritte Screen ist für die Darstellung der Kursentwicklung. |
|  | Muss | Funktional | Der vierte Screen soll mit zwei Punkten zeigen, dass meine App auf die Bewegungen des Handys reagiert. |
|  | Muss | Funktional | Die Liste auf dem ersten Screen beinhalten alle Anlageoptionen, die der User gespeichert hat. |
|  | Muss | Funktional | Die Liste kann vom Benutzer erstellt werden, indem er oder sie auf eine Anlageoption beim Suchscreen klickt. |
|  | Muss | Funktional | Auf der zweiten Seite sollen alle Anlageoptionen angezeigt werden, die von der API geholt wurden. |
|  | Muss | Funktional | Auf der zweiten Seite soll ein Eingabefeld sein, mit dem man nach Anlageoptionen suchen kann. |
|  | Muss | Funktional | Wenn man auf das Suchergebnis klickt, gelingt der User auf die dritte Seite, wo das Kursdiagramm ersichtlich ist. |
|  | Muss | Funktional | Der User kann Anlageoptionen aus der Liste löschen. |
|  | Muss | Funktional | Die Bewegungen des Handys sollen mit zwei Punkten gezeigt werden, die auf die Bewegungen reagieren. |
|  | Kann | Qualität | Die Bewegungen sollen mit einem roten und einem blauen Punkt angezeigt werden können. Der Rote bewegt sich horizontal und der Blaue vertikal. |
|  | Muss | Qualität | Die App soll responsive sein und sich an die aktuellen Handymodelle anpassen. |
|  | Muss | Qualität | Die App soll Kurse mit einem Diagramm korrekt illustrieren können. |
|  | Muss | Rand | Die App soll mit Expo erstellt werden |
|  | Muss | Rand | Die App soll in Visual Studio Code mit JavaScript programmiert werden. |

### Use Case-Diagramm

Ein Bild, das Text, Screenshot, Mond enthält.

Automatisch generierte Beschreibung

### Storyboard

Ein Bild, das Screenshot, Text, Grafikdesign, Design enthält.

Automatisch generierte Beschreibung

### Mockup

Wieso habe ich diese Navigation gewählt?

* Ich werden zum Text in der Navigation Icons benutzen, um die Navigation benutzerfreundlicher zu machen. Dies ist im Storyboard nicht ersichtlich.
* Ich habe diese Navigation gewählt, weil sie oft vorkommt und somit selbsterklären ist.
* Ich wollte kein Burgermenu machen, weil ich nur zwei direkte Navigationen gebrauche und weil sie nicht viel Platz auf dem Bildschirm einnehmen.

Wie reagiert meine App auf verschiedenen Bildschirmformate und Ausrichtungen, wieso habe ich diese Wahl getroffen?

* Meine App reagiert auf Ausrichtungen. Sie wechselt ihr Layout allerdings nicht beim Querformat.
* Ich habe die Ausrichtung auf diese Art und Weise programmiert, damit ich nicht mehr aufwand mit dem Design bringen muss und weil ein Querformat bei meiner App sinnlos ist und nicht notwendig, allerdings wollte ich eine Ausrichtung programmieren.
* Meine App ist responsive und passt sich an die unterschiedlichen Bildformate automatisch an.
* Ich wollte, dass meine App sich automatisch anpasst, um auch eine qualitativ gute App zu haben und um dem Benutzer das best-mögliche Erlebnis zu bringen.

Motion Sensor:

* Ich habe mir etwas Kreatives für meinen Ausrichtungssensor überlegt: Wenn man bei mir auf der App auf Motion unten in der Navigation klickt. Kommt man zu einem neuen Screen, der zwei Kreise anzeigt. Der Rote ist für die horizontalen Bewegungen und der Blaue für die Vertikale. Sie bewegen sich zu den Bewegungen des Handys.

### Usability-Konzepte und Vorgehen

1. Projektstruktur und Architektur:

*Verzeichnisstruktur:*

- /app

- Index.js

- SearchScreen.js

- MotionScreen.js

- /context

- WatchList

-/diagrams

- diagram.js

#### Architektur:

* **Komponentenbasierte Architektur:** Jeder Screen wird durch eine eigene Komponente repräsentiert.
* **Responsive Design:** Die Anpassung an verschiedene Bildschirmgrößen und -auflösungen wird durch CSS Flexbox oder Grid realisiert.

#### 2. Usability-Konzepte:

#### 2.1. Benutzerfreundlichkeit:

* **Intuitive Navigation:** Klare und leicht verständliche Navigation durch die drei Screens.
* **Konsistenz:** Einheitliches Design und Interaktionsmuster über alle Screens hinweg.

#### 2.2. Benutzereingabe:

* **Einfache Suche:** Klare Anzeige des Eingabefelds auf dem Suchscreen für die Suche nach Anlageoptionen.
* **Benutzerfreundliche Listenerstellung:** Die Möglichkeit, durch Klicken auf Suchergebnisse Anlageoptionen zur Liste auf dem ersten Screen hinzuzufügen.

#### 2.3. Visualisierung:

* **Aussagekräftige Diagramme:** Die Kursentwicklung wird durch gut lesbare Diagramme auf dem dritten Screen dargestellt.

#### 2.4. Interaktion:

* **Interaktive Elemente:** Benutzer kann durch Klicken auf Suchergebnisse und Anlageoptionen sowie durch Löschen von Anlageoptionen interagieren.
* **Feedback:** Rückmeldung über erfolgreiche Aktionen (z. B., wenn eine Anlageoption zur Liste hinzugefügt wurde).

#### 2.5. Qualität und Leistung:

* **Bewegungssensor-Integration:** Die App reagiert auf Bewegungen des Handys mithilfe eines Bewegungssensors.
* **Responsivität:** Die App passt sich dynamisch an verschiedene Handymodelle und Bildschirmgrößen an.

#### 3. Technische Umsetzung:

#### 3.1. Technologien:

* **Expo:** Verwendung von Expo für eine plattformübergreifende Entwicklung und einfachen Zugriff auf native Funktionen.
* **Visual Studio Code:** Die App wird mit JavaScript in Visual Studio Code entwickelt.

#### 3.2. Datenmanagement:

* **Zugriff auf Börsendaten:** Die ApiService-Komponente kümmert sich um den Zugriff auf Börsendaten über die API.
* **Datenpersistenz:** Verwendung von lokaler Speicherung oder einem geeigneten State-Management-Tool, um die API-Daten zu speichern

## Systemdokumentation

### Kurzanleitung

* + - 1. Die Dateien für die App wurden heruntergeladen.
      2. Extrahieren Sie die Dateien.
      3. Öffnen Sie das Projekt in Visual Studio Code.
         1. Dafür müssen Sie in den Ordner und danach auf den zweiten Ordner mit einem Rechtsklick.
         2. Wählen Sie Visual Studio Code, um den Ordner zu öffnen
      4. Wenn das Projekt geöffnet ist, müssen Sie sichergehen, dass Sie sich im letzten Petit-SiriguElisaLB-335 Ordner befinden
         1. Um das heraufzufinden, klicken Sie Ctrl. J, um ein Terminal zu öffnen.
         2. Schauen Sie sich als erstes den Pfad an, der Ihnen angezeigt wird. Wenn dort zwei Mal hintereinander Petit-SiriguElisaLB-335 steht, befinden Sie sich im korrekten Verzeichnis.
         3. Falls Sie sich nicht in diesem Verzeichnis befinden, geben Sie folgenden Befehl ein:  
            cd Petit-SiriguElisaLB-335
      5. Um die App zu starten, müssen Sie sich Expo Go auf ihrem Handy herunterladen.
      6. Zurück im Terminal geben Sie folgenden Befehl ein:  
         npm install
      7. Danach:  
         npx expo start
      8. Scannen Sie den QR-Code mit ihrer Handy Kamera. Sie werden automatisch zu Expo Go weitergeleitet und die App öffnet sich.

### Persistenzlösung

Ich habe mich für einen lokalen Speicher entschieden. Die synchronisierten daten von meiner API werden nur in eine Variable gespeichert und nach jedem öffnen vom Suchscreen neu geladen, um immer die Korrekten Kurse zu erhalten. Deshalb habe ich nicht dafür einen Speicher gebraucht.   
Ich brauche einen Speicher für die Liste, die der Benutzer erstellen kann. Damit diese auch nach dem Schliessen und erneutem Öffnen noch dieselbe ist. Meine Wahl fiel auf den lokalen Speicher, weil ich in meiner Recherche gelesen habe, dass sich diese Art gut für Mobile Applikationen eignet und weil ich bereits Erfahrungen mit solchen Speichern gemacht habe.

### Sensor

Ich habe eine Bewegungssensor in meiner App eingebaut. Ich wollte Etwas spannenderes machen, als die Ausrichtung des Bildschirms. Wenn man auf der Starseite bei der Navigation auf Motion klickt, kommt man auf einen fierten Screen der zwei Punkte anzeigt. Ein Punkt ist rot und der andere ist blau. Wenn man das Handy bewegt, bewegen sich auch die Punkte. Der Rote bewegt sich auf der Vertikalen und der Blaue auf der Horizontalen. Je nachdem wie das Handy ausgerichtet ist, kann man entweder beide oder keinen Punkt sehen.

### Netzwerkfunktionalität

In meiner App wird die Netzwerkfunktionalität beim Abrufen der Daten von der API über eine URL beansprucht. Ich fetche die Daten über eine URL, in der ein Key und Symbole enthalten sin. Der Key ist für den Zugriff auf die Daten und die Symbole für die Auswahl der Daten. Weil ich nicht alle Anlageoptionen fetchen wollte, habe ich mich für bestimmte entschieden, wofür ich die Symbole gebraucht habe. Der Server hat mir dann nur die ausgewählten Anlageoptionen zurückgegeben.

Wenn ich die Daten erhalten habe, speichere ich sie ab und zeige sie auf meinem zweiten Screen an. Durch das Suchen das Anlageoptionen werden diese Ergebnisse nach Symbolen geordnet und dem Benutzer angezeigt.

## Testdokumentation

### Testfälle

|  |  |  |  |
| --- | --- | --- | --- |
| **Nummer** | **Voraussetzung** | **Eingabe** | **Erwartete Ausgabe** |
|  | Die App ist geöffnet. | Keine, im Hintergrund: Die App versucht, auf die API zuzugreifen. | Die App zeigt aktuelle Börsendaten erfolgreich an. |
|  | Die App ist geöffnet. | Der Benutzer navigiert durch die App. | Die App zeigt vier Screens an: Liste, Suche, Motion und Kursentwicklung. |
|  | Die App funktioniert. | Benutzer startet die App. | Auf der Startseite ist eine Navigation vorhanden, mit der man zu allen Screens kommt. |
|  | Die App ist geöffnet. | Der Benutzer klickt auf Index. | Der erste Screen noch keine gespeicherten Anlageoptionen an. |
|  | Die App ist geöffnet. | Der Benutzer klickt auf Suchen. | Der zweite Screen zeigt ein Suchfeld an, in dem nach Anlageoptionen gesucht werden kann. |
|  | Die App ist geöffnet und der Such-Screen ist geöffnet. | Der Benutzer klickt auf eine Anlageoption. | Der dritte Screen zeigt die Kursentwicklung für ausgewählte Anlageoptionen an. |
|  | Die App ist geöffnet. | Der Benutzer klickt auf Motion. | Der vierte Screen zeigt zwei Punkte, die sich mit der Handybewegung mitbewegen. |
|  | Die App ist geöffnet und der Benutzer hat bereits Anlageoptionen gespeichert. | Der Benutzer navigiert zum ersten Screen. | Der erste Screen zeigt eine Liste aller vom Benutzer gespeicherten Anlageoptionen an. |
|  | Die App ist geöffnet und der Benutzer befindet sich auf dem zweiten Screen | Der Benutzer klickt auf eine Anlageoption. | Die Anlageoption wird automatisch bei der Liste auf dem ersten Screen gespeichert. |
|  | Die App ist geöffnet. | Der Benutzer klickt auf Suchen | Der zweite Screen zeigt alle ungefilterten Anlageoptionen an. |
|  | Die App ist geöffnet und zeigt den zweiten Screen an. | Der Benutzer schreibt "AP" in das Eingabefeld. | Die App zeigt Suchergebnisse, die mit AP anfangen an. |
|  | Die App ist geöffnet und zeigt Anlageoptionen auf dem zweiten Screen an. | Der Benutzer klickt auf eine Anlageoption. | Die App navigiert zum dritten Screen und zeigt das Kursdiagramm für die ausgewählte Anlageoption an. |
|  | Die App ist geöffnet und zeigt die Liste auf dem ersten Screen an. | Der Benutzer wählt eine Anlageoption aus und löscht sie. | Die ausgewählte Anlageoption wird aus der Liste entfernt. |
|  | Die App ist installiert und der vierte Screen ist geöffnet. | Das Handy wird vom Benutzer bewegt. | Die App reagiert korrekt auf die Bewegung des Handys. |
|  | Die App ist installiert und der vierte Screen ist geöffnet. | Das Handy wird vom auf und ab und von links nach rechts bewegt. | Zwei Punkte bewegen sich: Auf und ab: roter Punkt Links und rechts: blauer Punkt |
|  | Die App ist installiert und wird auf verschiedenen Handymodellen getestet. | Die App wird auf unterschiedlichen Handymodellen geöffnet. | Die App passt sich responsive an die Bildschirmgröße und -auflösung der verschiedenen Handymodelle an. |
|  | Die App ist auf dem Suchscreen geöffnet. | Der Benutzer klickt auf eine Anlageoption. | Die App zeigt Kurse mit einem Diagramm korrekt illustriert an. |

### Testprotokoll

**Testumgebung:** IPhone 13 pro

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nummer** | **Test Nr.** | **Datum** | **Resultat** | **Durchgeführt** |
|  | 1.1 | 17.12 | **OK** | Elisa Petit-Sirigu |
|  | 1.2 | 17.12 | **OK** | Elisa Petit-Sirigu |
|  | 1.3 | 17.12 | **OK** | Elisa Petit-Sirigu |
|  | 1.4 | 17.12 | **OK** | Elisa Petit-Sirigu |
|  | 1.5 | 17.12 | **OK** | Elisa Petit-Sirigu |
|  | 1.6 | 17.12 | **NOK** | Elisa Petit-Sirigu |
|  | 1.7 | 17.12 | **OK** | Elisa Petit-Sirigu |
|  | 1.8 | 17.12 | **NOK** | Elisa Petit-Sirigu |
|  | 1.9 | 17.12 | **NOK** | Elisa Petit-Sirigu |
|  | 1.10 | 17.12 | **OK** | Elisa Petit-Sirigu |
|  | 1.11 | 17.12 | **OK** | Elisa Petit-Sirigu |
|  | 1.12 | 17.12 | **NOK** | Elisa Petit-Sirigu |
|  | 1.13 | 17.12 | **NOK** | Elisa Petit-Sirigu |
|  | 1.14 | 17.12 | **OK** | Elisa Petit-Sirigu |
|  | 1.15 | 17.12 | **OK** | Elisa Petit-Sirigu |
|  | 1.16 | 17.12 | **OK** | Elisa Petit-Sirigu |
|  | 1.17 | 17.12 | **NOK** | Elisa Petit-Sirigu |

## Anhang

Quellen

Funktionalität:

* <https://medium.com/@awesome-jeremy/stocks-mobile-app-af13374c140b>
* <https://github.com/SridharanMurugadass/React-Native-Stock-App>

API-Wahl:

* <https://geekflare.com/de/best-stock-market-api/>
* <https://marketstack.com/quickstart>

Design:

* <https://www.youtube.com/watch?v=IhzrgITqOWE&ab_channel=SimonGrimm>

Sensoren:

* Rotation:

<https://docs.expo.dev/versions/latest/sdk/gyroscope/>

Bewegung:

<https://docs.expo.dev/versions/latest/sdk/devicemotion/>

* Lichtsensor:

<https://docs.expo.dev/versions/latest/sdk/light-sensor/>

Expo Router:

* <https://docs.expo.dev/router/navigating-pages/>