ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Konzeption und Realisierung einer mobilen Businessanwendung mit Unity

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Karolin Bartlmä
karolin.bartlmae@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Dr. Marc Schickler

2019

Fassung 30. April 2019

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Sowohl die Anzahl der Smartphones und damit die darauf installierten Anwendungen, als auch die Anzahl der Reisen steigt immer weiter an. Folglich ist es von immer größerer Wichtigkeit sich mit diesen Themen, auch in Kombination, auseinander zu setzen. Da das Smartphone unser alltäglicher Begleiter ist, sollte er uns auch bei unseren Reisen unterstützen, damit wir dort entspannen können. Für eine Reise müssen viele Dinge geplant werden, sodass es sinnvoll ist diese an einem Ort, beispielsweise in einer Anwendung, übersichtlich zu sammeln, auf diese jederzeit zugegriffen werden kann.

In dieser Arbeit wird eine Anwendung für das oben beschriebene Problem entwickelt. Das Ziel ist es hierbei eine leicht bedienbare Anwendung mit dem Programm Unity zu generieren, das für die Spielentwicklung vorgesehen ist. Zu zeigen ist nun, dass es ebenfalls möglich ist auch Businessanwendungen zu erstellen, die den gleichen Funktionsumfang besitzen wie in einem speziell dafür vorgesehenem Programm. Außerdem sollen Informationen und Funktionen miteinander kombiniert werden, die es auf diese Weise in keiner existierenden Anwendung gibt. Die Anwendung soll alle Reisen, Verbindungen, das Budget und die Packliste enthalten und die Möglichkeit besitzen die Route auf der Karte anzuzeigen.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während meines Studium begleitet und unterstützt haben.

Zunächst danke ich meinen Eltern, Michael und Michaela. Sie haben mir mein Studium ermöglicht und mich zu jedem Zeitpunkt unterstützt.

Mein besonderer Dank gilt Prof. Dr. Manfred Reichert, für die Begutachtung dieser Arbeit. Außerdem danke ich Dr. Marc Schickler, für die Unterstützung und Betreuung während der Bearbeitung.

Meinen Freunden, besonders Thuy und Michelle, danke ich für ihr Interesse und die Zeit, die sie in die Korrektur der Arbeit gesteckt haben.
Ich danke auch Stefanie, Susanne und Simon, die meine Idee immer unterstützt haben und die mir bei Problemen immer mit Rat zur Seite standen.

Inhaltsverzeichnis

1

Einleitung

Gerade im heutigen Zeitalter spielen Smartphones und die damit verbundenen Anwendungen eine wichtige Rolle. Genau deswegen ist es wichtig, sich näher mit diesen zu beschäftigen und dabei aktuelle Themen mit aufzugreifen. Wie man in Abbildung 1.1 erkennen kann, steigt die Anzahl der Smartphone-Nutzer immer weiter an und soll im Jahr 2020 2,87 Milliarden betragen. Somit lässt sich der stetig steigende Bedarf an Anwendungen für diese Smartphones und damit deren Notwendigkeit erklären.

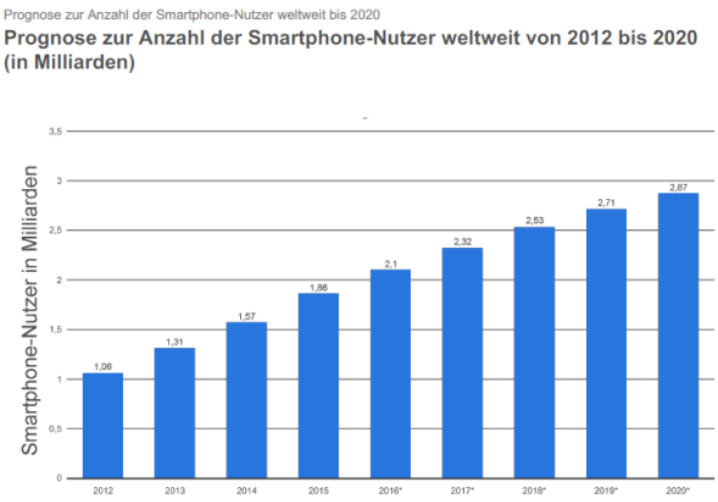

Abbildung 1.1: Prognose zur Anzahl der Smartphone-Nutzer weltweit von 2012 bis 2020. Laut dieser Prognose steigt die Anzahl der Smartphone-Nutzer auf 2,87 Milliarden [1]

Das Smartphone unterstützt uns heutzutage in vielen verschiedenen Lebensbereichen, wie Kommunikation und Informationsabruf. Aber nicht nur die grundlegenden Bedürfnisse sollen in der heutigen Zeit durch Handys abgedeckt werden, sondern darüber

hinaus auch andere Wünsche. Folglich wollen wir auch Anwendungen, die uns im Alltag unterstützen.

1.1 Problemstellung

Das Reisen stellt für viele einen bedeutenden Teil ihres Lebens dar und sollte aufgrund dessen von ihren ständigen Begleitern, den Smartphones, erleichtert werden. In Abbildung 1.2 erkennt man einen signifikanten Anstieg der Anzahl der Reisen, die von Deutschen unternommen werden. So liegt diese Zahl im Jahr 2018 bei ca. 70 Millionen Reisen.

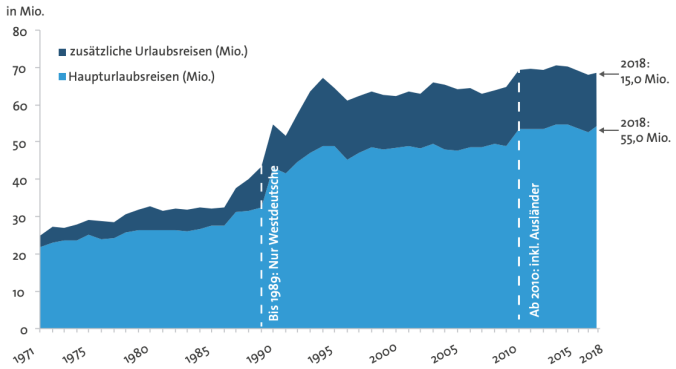

Abbildung 1.2: Volumen der Urlaubsreisen 1971 bis 2018 der deutschsprachigen Bevölkerung mit Alter 14+. Hier erkennt man den kontinuierlichen Anstieg seit 1971 [2]

Daher sollte auch dieser Bereich von einer Anwendung unterstützt werden. Es bestehen zwar bereits Applikationen in diesem Bereich, die meiner Meinung nach allerdings nicht den Umfang besitzen, den sie benötigen, um im angemessenen Rahmen zu helfen. Der Benutzer wird zwar in einem ausgewählten Punkt, wie etwa mit einer Packliste unterstützt, jedoch werden alle anderen relevanten Aspekte ausgeklammert.

1.2 Zielsetzung

Das Ziel dieser Bachelorarbeit ist es, eine Businessapplikation mithilfe des Programms Unity zu erstellen, die Jedermann auf ihren Reisen unterstützen soll. Im Rahmen dieser Bachelorarbeit wird eine Applikation erarbeitet, die es ermöglicht, auf alle relevanten Themen in einer Anwendung zuzugreifen. Des Weiteren wird eben diese Anwendung mithilfe eines Programms entwickelt, welches ursprünglich für Spiele konzipiert worden ist. Dies soll zeigen, dass auch Businessanwendungen gut erstellt werden können, wie es zum Beispiel mit Vergleichsprogrammen wie Android Studio möglich ist. Die erstellte Anwendung soll am Ende sowohl eine Packliste, einen Budgetplaner als auch eine Übersicht über alle Verbindungen, wie Züge, Flüge oder Busse, enthalten. Außerdem soll es möglich sein die Route über Google Maps anzeigen zu lassen. Die Packliste enthält alle wichtigen Utensilien, die für die jeweilige Reise von Bedeutung sind und kann durch die individuelle Anpassung von den verschiedensten Benutzern verwendet werden. Durch den Budgetplaner gibt es die Möglichkeit, sich vor der geplanten Reise darüber klar zu werden, wie viel Geld man dort zur Verfügung hat und wie viel man bereits ausgegeben hat. Darüber hinaus kann man auch alle Ausgaben auflisten, um einen Überblick zu erhalten, was man vielleicht noch kaufen, bzw. ob man im weiteren Verlauf sparen muss. Als letztes soll es möglich sein alle Verbindungen, die man benötigt, in einer übersichtlichen Aufstellung für jede Reise einzutragen. So kann jede Verbindung über ein neues Fenster eingetragen werden und wird dann auf der Startseite angezeigt. Um den Benutzer nicht zu viele Informationen auf einmal anzuzeigen, werden nur die wichtigsten Informationen auf der Hauptseite gezeigt. Möchte man nähere Informationen, kann man die gewünschte Reise auswählen und die restlichen Informationen werden dargestellt. Außerdem gibt es die Option, seine gesamte Reisestrecke auf Google Maps anzeigen zu lassen und somit auch alle weiteren Dienste von Google Maps nutzen zu können.

1.3 Struktur der Arbeit

Zu Beginn der Arbeit werden zunächst die Grundlagen in Kapitel 2 erläutert. Dabei wird erst auf die Entwicklungsumgebung Unity Abschnitt 2.1, anschließend auf das Datenformat JSON Abschnitt 2.2 und zuletzt auf Json.NET eingegangen. Im nächsten Kapitel 3 geht es um die Anforderungsanalyse und damit die funktionalen Abschnitt 3.2 und nicht-funktionalen Anforderungen Abschnitt 3.1. Daraufhin wird in Kapitel 4 das Konzept und der Entwurf der Anwendung aufgezeigt. In Kapitel 5 wird nun die tatsächliche Umsetzung gezeigt. Um nun einen Vergleich zu der zu Beginn erstellten Anforderungsanalyse zu haben, werden in Kapitel 6 die Unterschiede zur tatsächlich entstandenen Anwendung aufgezeigt. Damit man die Notwendigkeit dieser Anwendung gänzlich verstehen kann, werden nun in Kapitel 7 vergleichbare Applikationen betrachtet, um diese dann mit der hier Entstandenen zu vergleichen. Als letztes gibt es in Kapitel 8 einen Ausblick in die Zukunft.

2

Grundlagen

In diesem Kapitel werden zunächst die Grundlagen erläutert und wichtige Begrifflichkeiten zum Verständnis erklärt. Hierbei wird im Abschnitt 2.1 das Programm Unity vorgestellt, mit dem die Anwendung erstellt wurde. Anschließend wird das Datenformat JSON detaillierter besprochen, da so die Reisen gespeichert wurden. Zum Schluss wird das Framework Json.NET vorgestellt, da dies in der Anwendung verwendet wurde.

2.1 Unity

Unity ist eine Entwicklungsumgebung, die speziell für Spiele konzipiert ist. Allerdings können mit Unity auch jegliche Art von Anwendungen erstellt werden. So ist ein großer Vorteil von Unity, dass man hier für viele Plattformen, mit nur kleinen Änderungen, eine Anwendung erstellen kann. Folglich kann man beispielsweise schnell zwischen Android, IOS oder Windows wechseln. Außerdem ist Unity bekannt für seine Augmented und Virtual Reality Anwendungen, die unkompliziert erstellt werden können. Anwendungen in Unity werden üblicherweise mit C# umgesetzt und damit in der Entwicklungsumgebung Visual Studio [3]. Um dies zu verwirklichen, können in Unity sogenannte Skripte erstellt werden, die man dann in Visual Studio schreibt. Jedes dieser Skripte enthält zu Beginn zwei Methoden Start() und Update(), wobei Start() immer zu Beginn einmal aufgerufen wird und Update() für jeden Frame. Den grundsätzlichen Aufbau von Unity kann man in Abbildung 2.1 sehen. Dieser ist aufgeteilt in die Hierarchy, die Scene, den Inspector, das Project und das Game.

Dabei zeigt die *Hierarchy* eine bestimmte Szene der Anwendung oder des Spiels und die gesamten Objekte, die sich darin befinden. Diese Objekte nennt man in Unity

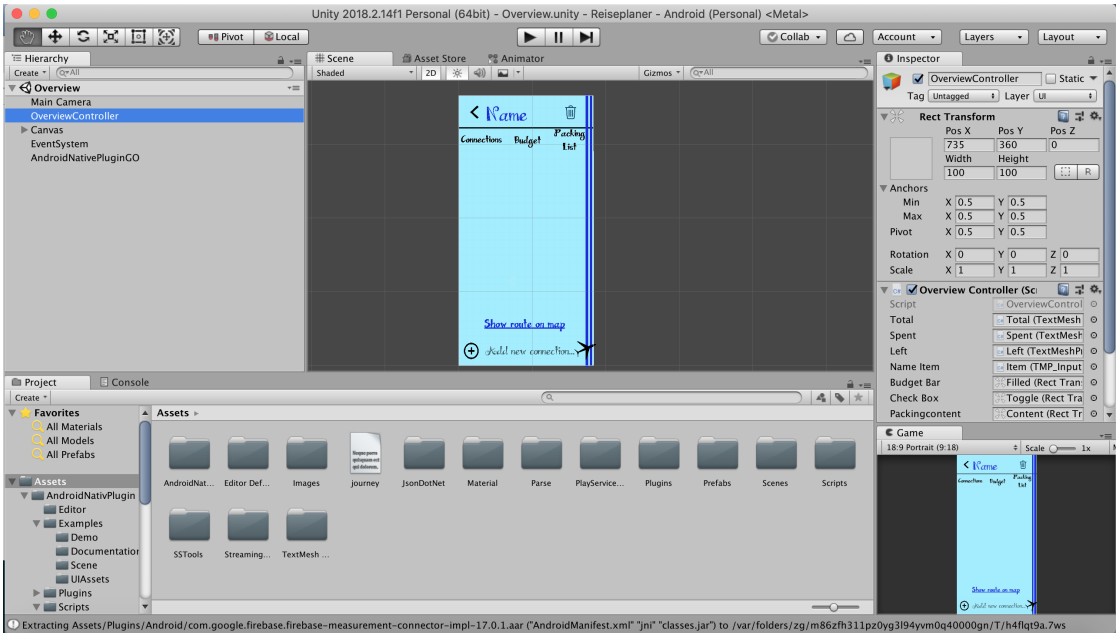

Abbildung 2.1: Der grundsätzliche Aufbau von Unity in Scene, Inspector, Project, Game und der Hierarchy

GameObjects und diese sind der Hauptbestandteil einer jeden Anwendung. So kann man diesen auch Skripte zuweisen, sodass diese bestimmte Dinge ausführen oder sie stellen einen Container für andere Objekte dar. Sie sind damit die Basisklasse einer Szene von Unity [3].

In der *Scene* erkennt man das vorläufige Aussehen der Anwendung in dieser Szene. Hier kann man auch die verschiedenen GameObjects verschieben, skalieren oder drehen und verschiedene Ansichten betrachten.

Der *Inspector* zeigt die detailliertere Ansicht eines GameObjects, also den Namen, ob es aktiv ist, Größe und alle weiteren hinzugefügten Eigenschaften. In Abbildung 2.1 sieht man z.B. das Gameobject OverviewController aktiv. Diesem wurde das passende Skript zugewiesen und die dafür benötigten weiteren Gameobjekte.

Beim *Project* sieht man alle verwendeten Assets des Projekts. Dazu gehören die gesamten Skripte, Szenen, Frameworks, Bilder oder heruntergeladene Assets vom Assetstore. Das bedeutet, dass in Unity die Möglichkeit besteht, bereits vorgefertigte Assets herunterzuladen, damit man diese nicht mehr selbstständig erstellen muss. Ein Beispiel, das

man hier sieht wäre TextMeshPro, mit dem man verschiedene Texte oder Eingabefelder erstellen kann.
Als letztes kann man noch in der unteren, rechten Ecke das *Game* erkennen. Hier sieht man eine Vorschau davon, wie die Anwendung dann im geforderten Format aussieht.
So ist es insgesamt gut möglich eine Anwendung zu konstruieren, die den Nutzer anspricht und die auch eine gute Performance hat.

2.2 JSON

JSON (Javascript Object Notation) ist ein Datenformat mithilfe dessen man Daten zwischen Anwendungen austauschen kann oder Daten gespeichert werden können. Grundsätzlich ist JSON sprachenunabhängig, basiert jedoch auf der Programmiersprache Javascript. Generell besteht eine .json-Datei entweder aus mehreren name/value Paaren oder einer Liste aus verschiedenen Werten [4]. In Abbildung **??** erkennt man eine einfache Struktur, die eine Person mit den jeweiligen Eigenschaften und deren Kindern darstellt. Eine .json-Datei beginnt immer mit einer geschweiften Klammer und zeigt damit an, dass nun ein Objekt folgt. In diesem Fall ist dies eine Person mit den Paaren name, alter, verheiratet, beruf und kinder und deren jeweiligen Werten. Die Kinder werden hier in einem Array (eckige Klammern) dargestellt, da man oftmals mehr als ein Kind hat. Ein Kind ist hierbei wieder ein eigenes Objekt mit name/value Paaren.

```
1 {
2   "name": "Georg",
3   "alter": 47,
4   "verheiratet": false,
5   "beruf": null,
6   "kinder": [
7     {
8           "name": "Lukas",
9           "alter": 19,
```

```
10            "schulabschluss": "Realschule"
11          },
12          {
13            "name": "Lisa",
14            "alter": 14,
15            "schulabschluss": null
16          }
17    ]
18  }
```

Listing 2.1: Beispiel einer JSON-Struktur, die eine Person mit seinen Eigenschaften und deren Kindern darstellt

2.3 Json.NET

Json.NET ist ein Framework von Newtonsoft, mit dem man Objekte in JSON serialisieren oder JSON in Objekte deserialisieren lassen kann. Außerdem kann auch LINQ in JSON umgewandelt werden. Darüber hinaus kann so leicht überprüft werden, ob es sich um valides JSON handelt oder ob man doch vorher einen Fehler gemacht hat. Für Unity gibt es auch ein Asset, das man herunterladen und mithilfe dessen Json.NET auch hier verwendet werden kann. So können vorher Klassenstrukturen angelegt werden, die dann mit Json.NET einfach in JSON-Format umgewandelt und gespeichert werden können. Alle Attribute, die dabei als public deklariert wurden, werden dann in die Datei gespeichert. Dies ermöglicht eine leichte und persistente Erhaltung der Daten einer Anwendung. Außerdem ist auch die Performance, siehe Abbildung 2.2, nicht zu vernachlässigen, die sowohl schneller als der DataContractJsonSerializer als auch der JavaScriptSerializer ist [5].

Im folgenden Quellcode sieht man die Anwendung von Json.NET in Unity in der Programmiersprache C#. Hier wird eine Liste mithilfe von Json.NET in ein JSON-Objekt verwandelt und anschließend in eine Datei geschrieben.

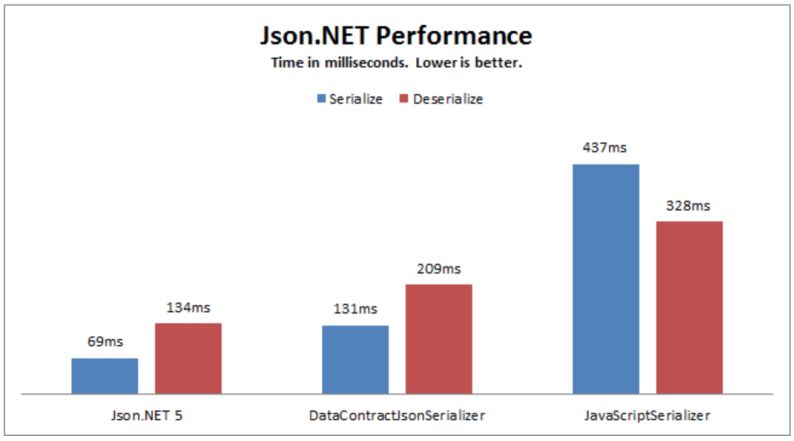

Abbildung 2.2: Performancevergleich von Json.Net mit DataContractJsonSerializer und JavaScriptSerializer [5]

```
1  public void SaveJourney (List<Journey> journeys) {
2          var setting = new JsonSerializerSettings();
3          setting.Formatting = Formatting.Indented;
4
5          var json = JsonConvert.SerializeObject(journeys,
6           setting);
7          var path = Path.Combine(Application.persistentDataPath,
8           "journey.json");
9
10         File.WriteAllText(path, json);
11
12     }
```

Listing 2.2: Beispiel Json.NET

3

Anforderungsanalyse

Die Anforderungsanalyse ist ein wichtiger und zentraler Punkt in der Softwareentwicklung. Das Ziel ist es hierbei, die vom Kunden gewünschten Softwareanforderungen festzulegen, wobei diese Anforderung einem qualitäts- oder funktionsbezogenem Merkmal der Software entspricht. Häufig entstehen Fehler bereits dadurch, dass man hier die Anforderungen nicht richtig definiert und so im weiteren Verlauf nicht zu dem Ergebnis gelangt, das ursprünglich geplant war. Diese Fehlerkategorie kann bis zu 50% der gesamten Fehler ausmachen und oftmals treten diese Fehler erst im späteren Verlauf der Implementierung auf. Deshalb ist es wichtig schon hier genau zu arbeiten und die Anforderungen treffend zu definieren [6]. Zusammengefasst soll in der Anforderungsanalyse eine vollständige Beschreibung dessen, was ein System tun soll, nicht aber wie es das tun soll, stehen. [7]. Unterschieden wird hierbei in funktionale und nicht-funktionale Anforderungen, wie man in Abbildung 3.1 nochmals deutlich sehen kann.

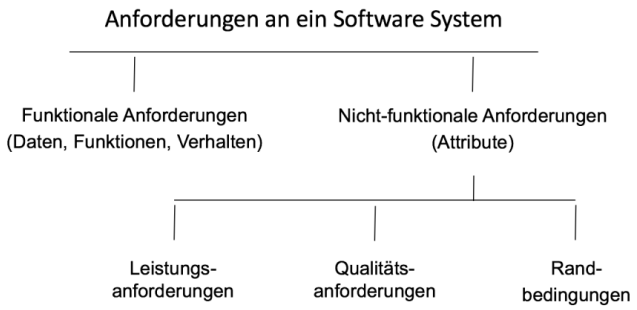

Abbildung 3.1: Unterscheidung der Anforderungen in funktionale und nicht-funktionale Anforderungen [7]

Im Folgenden werden zuerst die nicht-funktionalen und anschließend die funktionalen Anforderungen der Anwendung definiert.

3.1 Nicht-funktionale Anforderungen

In Abbildung 3.1 kann man bereits erkennen, dass nicht-funktionale Anforderungen auch Attribute genannt werden können. Gemeint ist damit, in welcher Qualität die Anwendung arbeiten soll [7].

ID	Q01
Titel	Dokumentation
Beschreibung	Mindestens 80 Prozent der implementierten Funktionen sollen dokumentiert werden.

ID	Q02
Titel	Verfügbarkeit
Beschreibung	Die Anwendung muss korrekt auf zulässige Benutzereingaben reagieren. Auf Benutzereingaben muss bei der nächsten Bildwiederholung reagiert werden.

ID	Q03
Titel	Zeitverhalten
Beschreibung	Das System muss innerhalb von 10ms das Ergebnis berechnet haben.

ID	Q04
Titel	Robustheit
Beschreibung	Die Anwendung darf nicht abstürzen. Bei 100 Spielen darf maximal 1 Spiel aufgrund eines Fehlers abgebrochen werden.

ID	Q05
Titel	Bedienbarkeit
Beschreibung	Die Anwendung muss leicht und verständlich zu bedienen sein. Auf Benutzereingaben muss die Anwendung richtig reagieren.

3.2 Funktionale Anforderungen

Auch die funktionalen Anforderungen stellen wie in Abbildung 3.1 schon beschrieben, Daten, Funktionen und Verhalten dar. So werden hier alle Dinge definiert, die die Anwendung leisten, also alle Funktionen, die das System haben soll [8].

ID	FA01
Titel	Nutzer
Beschreibung	Der Nutzer steuert die gesamte Anwendung und kann neue Elemente hinzufügen.
Begründung	Um dem Nutzer die Handhabung der Anwendung zu ermöglichen.
Abhängigkeit	Alle Anforderungen
Prioritäten	++

ID	FA02
Titel	Reisen
Beschreibung	Der Nutzer hat die Möglichkeit mehrere Reisen mit Name, Datum und Bild zu erstellen.
Begründung	Dies ermöglicht dem Nutzer eine Übersicht über seine geplanten Reisen zu erlangen.
Abhängigkeit	-
Prioritäten	++

ID	FA03
Titel	Budget
Beschreibung	Jede Reise hat ein individuelles Budget, das der Nutzer festlegen kann.
Begründung	Dies ermöglicht es dem Nutzer sein verfügbares Geld für jede Reise einzuplanen.
Abhängigkeit	FA02
Prioritäten	++

ID	FA04
Titel	Budget insgesamt
Beschreibung	Der Nutzer kann ein Gesamtbudget für die Reise festlegen, welches ihm daraufhin in der Anwendung gezeigt wird.
Begründung	Damit der Nutzer weiß, wie viel Geld er insgesamt zur Verfügung hat.
Abhängigkeit	FA03
Prioritäten	+

ID	FA05
Titel	Budget Ausgaben
Beschreibung	Der Nutzer kann jede seiner Ausgaben in der Anwendung speichern. Diese zeigt ihm das verfügbare Restbudget und alle Ausgaben übersichtlich an.
Begründung	Der Nutzer kann so genau verfolgen für was er bereits Geld ausgegeben hat.
Abhängigkeit	FA03, FA04
Prioritäten	+

ID	FA06
Titel	Budgetrest
Beschreibung	In der Anwendung soll dem Nutzer angezeigt werden, wie viel Geld er noch zur Verfügung hat.
Begründung	Dies ermöglicht es dem Nutzer seine weiteren Ausgaben zu planen.
Abhängigkeit	FA03, FA04, FA05
Prioritäten	+

ID	FA07
Titel	Packliste
Beschreibung	Für jede Reise soll es auch eine Packliste geben, die der Nutzer individuell erweitern kann.
Begründung	Der Nutzer kann so vor der Reise packen und abhaken, welche Items er bereits eingepackt hat.
Abhängigkeit	FA02
Prioritäten	++

ID	FA08
Titel	Packliste Kategorie
Beschreibung	Für die Packliste soll es verschiedene Kategorien geben, zu denen die Items gehören.
Begründung	Dies erhöht die Übersichtlichkeit für den Nutzer und er muss nicht lange suchen.
Abhängigkeit	FA07
Prioritäten	0

ID	FA09
Titel	Packliste Items
Beschreibung	In jeder Packliste gibt es verschiedene Items, die der Nutzer abhaken kann. Es ist außerdem möglich weitere Items hinzuzufügen, falls man spezielle Sachen benötigt. Jedes dieser Items gehört einer Kategorie an.
Begründung	Damit der Nutzer einzelne Sachen abhaken kann und somit weiß was schon gepackt ist.
Abhängigkeit	FA07, FA08
Prioritäten	+

ID	FA10
Titel	Verbindungen
Beschreibung	Zu jeder Reise kann der Nutzer beliebig viele Verbindungen hinzufügen. Dazu gehören Züge, Busse und Flugzeuge.
Begründung	Mit den Verbindungen erhält der Nutzer eine Übersicht, an welchem Ort und zu welchem Zeitpunkt die Reise stattfindet.
Abhängigkeit	FA02
Prioritäten	++

ID	FA11
Titel	Züge
Beschreibung	Eine der Verbindungsarten ist der Zug. Zu dieser Verbindung kann der Nutzer das Datum, die Zeit, das Gleis, den Wagon, den Platz, die Klasse, den Abfahrts- und Ankunftsort hinzufügen.
Begründung	Der Zug ist einer der gängigsten Verbindungstypen und muss dem Nutzer zur Verfügung stehen.
Abhängigkeit	FA10
Prioritäten	+

ID	FA12
Titel	Bus
Beschreibung	Eine der Verbindungsarten ist der Bus. Zu dieser Verbindung kann der Nutzer das Datum, die Zeit, den Platz, den Abfahrts- und Ankunftsort hinzufügen.
Begründung	Der Bus ist eine wichtige Verbindungsart und bietet dem Nutzer die Möglichkeit sich zu entscheiden.
Abhängigkeit	FA10
Prioritäten	+

ID	FA13
Titel	Flugzeug
Beschreibung	Eine der Verbindungsarten ist das Flugzeug. Zu dieser Verbindung kann der Nutzer das Datum, die Zeit, das Gate, die Airline, die Flugnummer, den Ankunfts- und Abflugsort hinzufügen.
Begründung	In weiter entfernte Länder ist ausschließlich diese Verbindungsart möglich.
Abhängigkeit	FA10
Prioritäten	+

ID	FA14
Titel	Weitere Verbindungen
Beschreibung	Dies ist ein Link/Button, der es dem Nutzer ermöglicht weitere Züge für diesen Abfahrtsort und die angegebene Uhrzeit anzuzeigen.
Begründung	Dies ermöglicht es dem Nutzer nach Alternativen zu schauen, um eventuell früher zu verreisen oder nach einem neuen Zug zu schauen, falls der andere verpasst wurde.
Abhängigkeit	FA10, FA11
Prioritäten	0

ID	FA15
Titel	Auf der Karte anzeigen
Beschreibung	Ermöglicht es dem Nutzer seine angegebene Strecke in Google Maps nachzuvollziehen.
Begründung	Bietet dem Nutzer die Möglichkeit die Verbindung besser zu verstehen und auch die Features von Google Maps zu nutzen.
Abhängigkeit	FA10, FA11, FA12, FA13
Prioritäten	+

ID	FA16
Titel	Hauptbildschirm
Beschreibung	Auf diesem Bildschirm sollen dem Nutzer alle kommenden und kürzlich vergangenen Reisen angezeigt werden. Außerdem kann er hier mit einem Plus-Symbol neue Reisen erstellen.
Begründung	Dieser zeigt dem Nutzer übersichtlich alle Reisen, damit er immer alles auf einem Blick hat.
Abhängigkeit	FA02
Prioritäten	+

ID	FA17
Titel	Übersicht Budget, Packliste, Verbindungen
Beschreibung	Dies ist ein Bildschirm bei dem man über drei Tabs verschiedene Übersichten erhält. In der Budgetübersicht sieht man alle seine Ausgaben, das Gesamtbudget und wie viel noch übrig ist. In der Packliste sieht man eine Übersicht über die bereits gepackten Gegenstände. Bei dem Tab zu Verbindungen sieht man alle eingegebenen zeitlich sortiert untereinander. Oben soll der Name der gewählten Reise sehen.
Begründung	Dies soll dem Nutzer eine einfache und übersichtliche Handhabung ermöglichen.
Abhängigkeit	FA02, FA03, FA04, FA05, FA06, FA07, FA08, FA09, FA10, FA11, FA12, FA13, FA14, FA15, FA16
Prioritäten	+

ID	FA18
Titel	Detailansicht Verbindung
Beschreibung	Für jede Verbindung sollte es möglich sein, eine detailliertere Ansicht zu erhalten, wenn dies der Nutzer möchte.
Begründung	Übersichtliche Darstellung, da nicht alle Informationen auf der Verbindungsübersicht angezeigt werden.
Abhängigkeit	FA10, FA11, FA12, FA13, FA17
Prioritäten	0

ID	FA19
Titel	Reisen verändern
Beschreibung	Es soll die Möglichkeit geben, eine bereits existierende Reise hinsichtlich ihres Namens, ihres Bildes oder ihres Datums zu verändern.
Begründung	Der Nutzer soll die Möglichkeit besitzen, bei Planänderungen oder falschen Eingaben, die Reise im Nachhinein noch zu verändern.
Abhängigkeit	FA02
Prioritäten	0

ID	FA20
Titel	Packliste Kategorie erstellen
Beschreibung	Der Nutzer soll der Packliste eigene Kategorien hinzufügen können, denen anschließend Items zugewiesen werden.
Begründung	So hat der Nutzer die Möglichkeit die Packliste nach seinen Wünschen individuell zu erweitern.
Abhängigkeit	FA08
Prioritäten	-

ID	FA21
Titel	Reise löschen
Beschreibung	Der Nutzer hat die Möglichkeit eine erstellte Reise wieder zu löschen.
Begründung	Der Nutzer kann so seine Anwendung übersichtlich halten und wieder Speicherplatz hinzugewinnen.
Abhängigkeit	FA02
Prioritäten	0

ID	FA22
Titel	Verbindung löschen
Beschreibung	Der Nutzer hat die Möglichkeit eine erstellte Verbindung wieder zu löschen.
Begründung	Der Nutzer kann so eine unnötige oder bereits vergangene Verbindung wieder löschen.
Abhängigkeit	FA10
Prioritäten	0

ID	FA23
Titel	Packliste Item löschen
Beschreibung	Der Nutzer hat die Möglichkeit ein Item das in der Packliste existiert zu löschen.
Begründung	Dies ermöglicht es dem Nutzer Items zu löschen, die er nicht benötigt.
Abhängigkeit	FA07, FA09
Prioritäten	0

ID	FA24
Titel	Packliste Kategorie löschen
Beschreibung	Der Nutzer hat die Möglichkeit eine Kategorie aus der Packliste zu löschen.
Begründung	Dies ermöglicht es dem Nutzer Kategorien zu löschen, die er nicht benötigt.
Abhängigkeit	FA07, FA08
Prioritäten	-

ID	FA25
Titel	Budget Ausgabe löschen
Beschreibung	Der Nutzer hat die Möglichkeit eine Ausgabe aus seinem Budget zu löschen.
Begründung	Dies ermöglicht es dem Nutzer Ausgaben zu löschen, die er doch nicht gebraucht hat.
Abhängigkeit	FA03, FA05
Prioritäten	0

4

Konzept und Entwurf

In Kapitel 4 sollen das grundsätzliche Konzept und ein Entwurf für die spätere Realisierung vorgestellt werden. Dazu wird zunächst das Entity-Relationship-Diagramm zur Anwendung in Abschnitt 4.1 erläutert. Anschließend werden in Abschnitt 4.3 die Mockups der Anwendung gezeigt, um einen ersten Eindruck zur Anwendung zu erhalten. Als letztes wird auf die lokale Persistenz in Abschnitt 4.4 eingegangen.

4.1 Entity-Relationship-Diagramm

Ein ER-Diagramm (Entity-Relationship-Diagramm) dient oftmals als Grundlage für die spätere Anwendung oder Software und zählt damit als ein wichtiger Punkt in der Softwareerstellung. So werden mit einem ER-Diagramm Objekte typisiert und deren Beziehung zu anderen Objekten oder Attributen dargestellt [9]. In Abbildung 4.1 erkennt man das ER-Diagramm zur Anwendung.

Das zentrale Objekt ist die *Reise*, die deswegen auch in der Mitte angeordnet ist. Sie hat Beziehungen zu fast allen anderen Objekten, so besitzt eine Reise immer eine Packliste, ein Budget und mehrere Verbindungen (Flüge, Busse, Züge). Darüber hinaus ist sie mit dem Benutzer verbunden, der die Reise erstellt hat. Außerdem gehören zu einer Reise auch immer ein Datum, ein Bild und ein Name, die der Benutzer selber wählen kann.
Die *Packliste* gehört umgekehrt erwartungsgemäß zu einer Reise, hat abgesehen davon aber noch mehrere Objekte, die in der Packliste stehen.
Das *Objekt* meint hier ein einzelnes Objekt innerhalb der Packliste. Dieses besitzt außerdem einen Namen wie Ausweis und das Attribut „Schon eingepackt“ , was hier bedeutet,

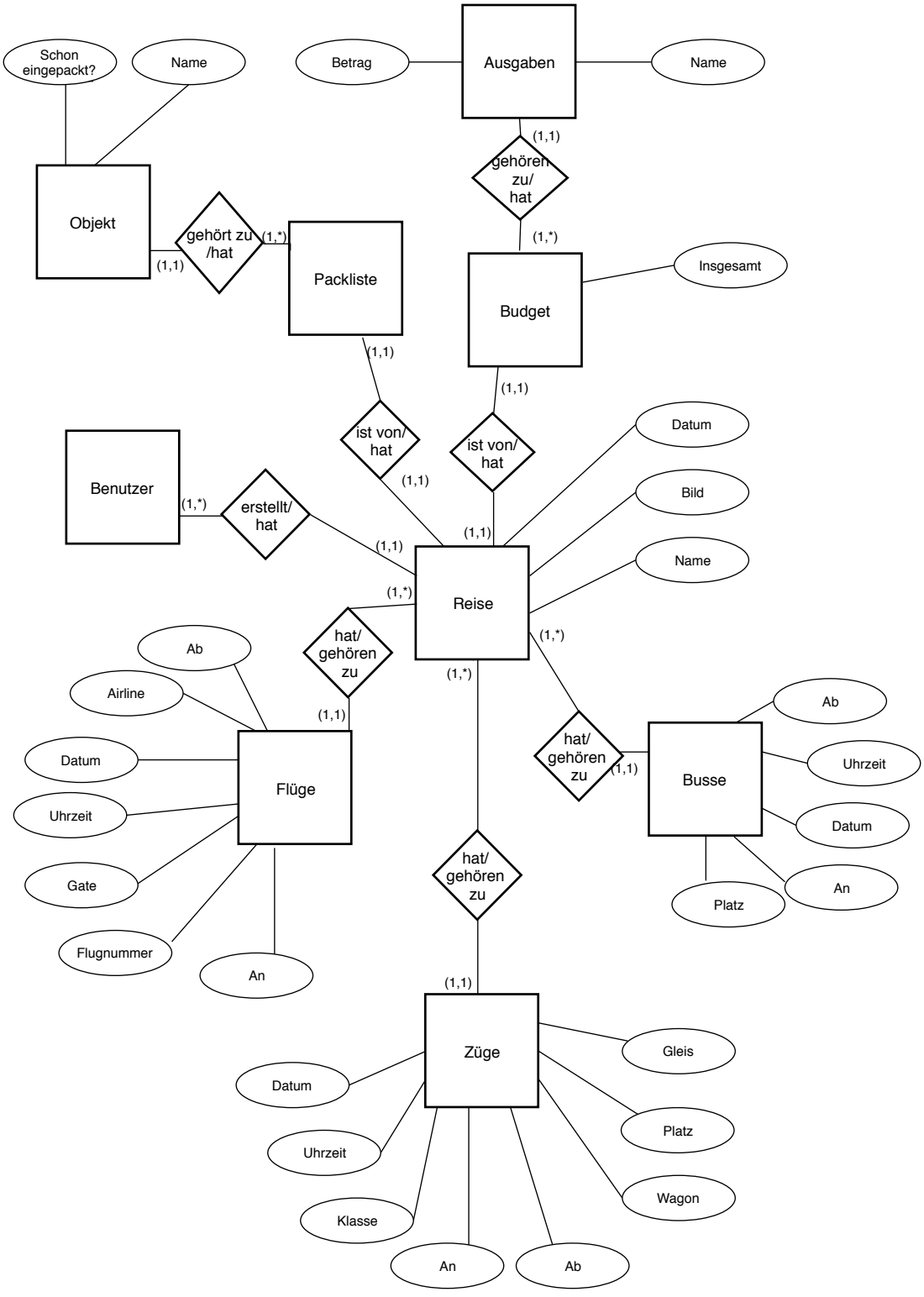

Abbildung 4.1: Entity-Relationship-Diagramm zur Anwendung. Die Entitäten Reise, Busse, Flüge, Züge, Benutzer, Budget, Ausgaben und Objekt werden zueinander in Relation gesetzt und die jeweiligen Attribute zugeordnet.

ob das Objekt schon in den Koffer gepackt wurde und damit bereits abgehakt ist.
Auch das *Budget* gehört zur Reise, aber kann abgesehen davon auch mehrere Ausgaben besitzen. Des Weiteren hat das Budget auch das Attribut „Insgesamt“ , welches angibt, wie viel Geld man insgesamt zur Verfügung hat.
Die *Ausgaben* selbst haben auch noch zwei weitere Attribute „Name“ und „Betrag“ .
Die Objekte, die zusammengefasst die Verbindungen darstellen sind *Flüge*, *Busse* und *Züge*. Diese gehören jeweils beliebig oft zu einer Reise und besitzen alle mehrere Attribute. So hat das Objekt Busse noch die Attribute „Ab“ , „Uhrzeit“ , „Datum“ , „An“ und „Platz“ , damit dieser eindeutig definiert ist und alle wichtigen Informationen beinhaltet. Auch das Objekt Züge hat die gleichen Attribute. Zusätzlich dazu gibt es hier noch das Attribut „Gleis“ , „Wagon“ und „Klasse“ . Für die Flüge wurden die selben Attribute gewählt, wie bei den Bussen, nur dass hier noch die Flugnummer, das Gate und die Airline hinzukommen.

4.2 Model-View-Controller

Das Model-View-Controller (MVC) Pattern unterteilt eine Anwendung in das Model, die View und den Controller, um diese zu strukturieren. In Abbildung 4.2 erkennt man die klare Aufteilung in die drei Komponenten und deren Beziehungen untereinander.

Das Model beschreibt hierbei die grundsätzliche Struktur der Anwendung, die View enthält nur graphische Komponenten und erhält dabei ihre Informationen vom Model und der Controller steuert die Anwendung [11]. Auch diese Anwendung wurde nach dem MVC Pattern aufgebaut. Dabei stützt sich das Model weitgehend auf dem ER-Diagramm aus dem vorherigen Abschnitt. So ist eine Reise in ihre Attribute Name, Date und PicturePath und die Objekte mit denen sie verbunden ist, PackingList, Budget, Airplane, Bus und Train aufgeteilt. Diese stellen die Variablen der Klasse Journey dar.

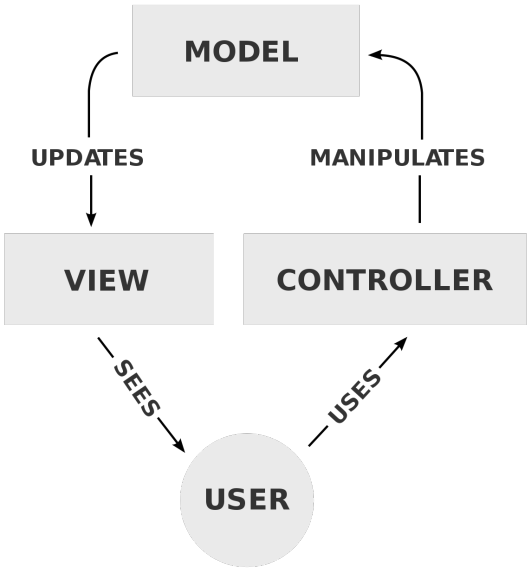

Abbildung 4.2: Das Model-View-Controller Pattern [10]

```
1  public class Journey
2  {
3      public string Name { get; set; }
4      public string Date { get; set; }
5      public string PicturePath { get; set; }
6      public PackingList PackingList { get; set; }
7      public Budget Budget { get; set; }
8      public List<Airplane> Airplane { get; set; }
9      public List<Bus> Bus { get; set; }
10     public List<Train> Train { get; set; }
11 }
```

Listing 4.1: Klasse Reise im Model

Genauso verhält es sich für alle anderen Objekte aus dem ER-Diagramm. Folglich besteht das Model aus den Klassen Journey, Bus, Airplane, Train, Budget und PackingList. Die View stellt in Unity die Gameobjects dar und ist deswegen sowieso getrennt vom Rest. Die restlichen Skripte stellen den Controller dar und führen damit Tätigkeiten, wie das Öffnen einer neuen Szene, aus.

```
1 public void Open () {
2         TextMeshProUGUI text =
3         GetComponentInChildren<TextMeshProUGUI>();
4         PlayerPrefs.SetString("journeyclicked", text.text);
5         SceneManager.LoadScene(SceneManager
6         .GetActiveScene().buildIndex + 1);
7     }
```

Listing 4.2: Öffnen einer neuen Szene nach Klick auf Reise

4.3 Mock-ups

Um einen Überblick über die Funktionen und die Aufteilung in der Anwendung zu erhalten, wurden zunächst Mock-ups zu den wichtigsten Bildschirmen erstellt. Diese wurden mit der Internetseite NinjaMock [12] kreiert, mithilfe deren es einfach ist, klickbare Mock-ups zu designen. Grundsätzlich lässt sich über die Gestaltung sagen, dass der Hintergrund einen hellen Blauton hat und die meisten weiteren Elemente einen verwandten Farbton besitzen. Außerdem ist auf jedem Bildschirm ein kleines Flugzeug zu erkennen, welches auch für das Icon verwendet wird. Den Rand zieren zwei blaue Streifen, die an den Hintergrund angepasst sind. In Abbildung 4.3 erkennt man den Hauptbildschirm, der direkt nach dem Start der Anwendung erscheint. Man erkennt hier die Reisen, die bald stattfinden und welche die schon vergangen sind. Jede Reise hat hier einen individuellen Farbverlauf und enthält die Informationen Name und Datum. Am unteren Rand des Bildschirms ist ein Plus angeordnet, mit dem man neue Reisen erstellen kann. In Abbildung 4.4 sieht man die Übersichtsseite über die eingetragenen Verbindungen. Zu dieser Seite gelangt man über einen Klick auf eine bestimmte Reise

Abbildung 4.3: Hauptseite der Anwendung mit Übersicht über alle Reisen

und wird dementsprechend angezeigt. In der obersten Spalte erkennt man den Namen der Reise und einen Zurück-Pfeil, der zurück zur Hauptseite führt. Darunter sind drei Tabs angeordnet, die man auswählen kann. Ist das Wort Connections unterstrichen, kann man erkennen, dass gerade die Verbindungen angezeigt werden. Zur Auswahl stehen hier außerdem die beiden Tabs Budget und Packing List. Da wir uns in der Verbindungsübersicht befinden, sieht man hier auch als Beispiel eine Flug- und eine Busverbindung. Eine Verbindung besteht jeweils aus einem Bild, welches die Art der Verbindung verdeutlicht. Dieses Bild ist bei jedem Verbindungselement an der linken Seite angeordnet. Daneben steht der Abfahrts- und Ankunftsort und direkt darunter das Datum und die Uhrzeit. Darüber hinaus wird bei den Zügen zusätzlich die Option „further connection... “ angezeigt, mit der man weitere Züge zu dieser Zeit suchen kann. Im unteren Bereich des Bildschirm erkennt man erneut das Plus-Zeichen, mit dem man auch hier eine neue Verbindung hinzufügen kann. Als letztes ist unter dem Plus ein

Abbildung 4.4: Übersichtsseite der eingetragenen Verbindungen der Reise

Link mit „Show route on map “, der die Route in Google Maps zeigen soll. Als nächstes erkennt man in Abbildung 5.9 die Übersichtsseite für das Budget. Diese entspricht dem zweiten Tab der allgemeinen Übersichtsseite. Im Zentrum steht hier ein Kreisdiagramm, das übersichtlich das noch verfügbare Budget zeigt. In Blau wird das noch verfügbare Geld, in Orange das ausgegebene Geld dargestellt. Darunter wird in den selben Farben noch einmal das verfügbare, das ausgegebene und das gesamte Geld in Ziffern angezeigt. Das gesamte, zur Verfügung stehende Geld ist in Grün unterlegt, da Grün mit noch verfügbarem Geld assoziiert wird. Darunter, mit einem Strich abgetrennt, sind darunter alle Ausgaben aufgelistet, die jeweils einen Namen und einen Betrag haben und über das Plus ganz unten hinzugefügt werden können. Die nächste Abbildung 4.6 zeigt den letzten Tab und damit die Packliste der Reise an. Diese ist in verschiedene Kategorien aufgeteilt und jede dieser Kategorien beinhaltet Checkboxen mit den Namen der Objekte. Auch hier ist im unteren Bereich ein Plus angeordnet, mit dem man neue Objekte erstellen kann. In Abbildung 4.7 sieht man den Bildschirm, nachdem man bei

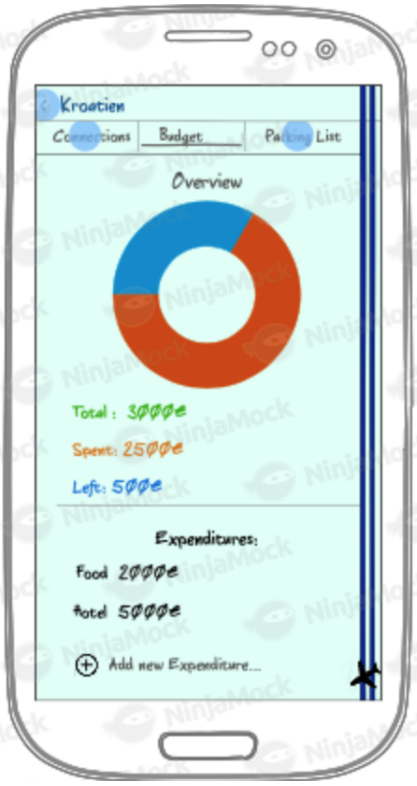

Abbildung 4.5: Übersicht über alle Ausgaben und das gesamte Budget der Reise

der Verbindungsübersicht auf das Plus-Zeichen geklickt hat. Es erscheint ein Screen, bei dem man nun zunächst eine Verbindungsart auswählen muss. Auch hier hat man natürlich die Möglichkeit über den Zurück-Pfeil wieder zur Übersicht zu gelangen. Möchte man allerdings eine neue Verbindung erstellen, hat man die Möglichkeit zwischen einer Zug-, Bus-, oder Flugverbindung zu wählen. Diese werden mit dem jeweiligen Bild gekennzeichnet und durch einen Klick darauf ausgewählt. Als letztes Mock-up zeigt die Abbildung 4.8 das Erstellen einer neuen Verbindung. Wie in den meisten Bildschirmen hat man auch hier die Möglichkeit über einen Zurück-Pfeil einen Schritt zurück zu gehen und damit eine andere Verbindungsart auszuwählen. Dieser Bildschirm besteht hauptsächlich aus Eingabefeldern, bei denen man die nötigen Informationen zu dieser Verbindung eintragen kann. Ist man damit zufrieden, kann man die Eingabe mit Klick auf den Okay-Button rechts unten bestätigen. Möchte man doch keine Verbindung erstellen, kann man das Ganze mit Klick auf den Cancel-Button abbrechen.

Abbildung 4.6: Anzeige der Packliste mit den verschiedenen Kategorien

Insgesamt wurden in den Mock-ups die wichtigsten, aber nicht alle, Funktionalitäten der Anwendung aufgegriffen, um einen ersten Eindruck und Designentwurf zu erlangen. In Kapitel 5 werden dann alle Bildschirme in der endgültigen Version gezeigt.

Abbildung 4.7: Auswahl der Art der hinzuzufügenden Verbindung: Flug-, Bus- oder Zugverbindung

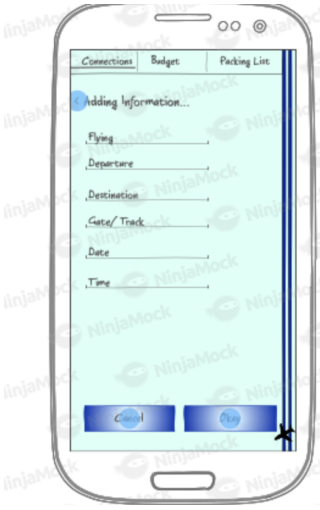

Abbildung 4.8: Hinzufügen einer neuen Verbindung (hier: Flug)

4.4 lokale Persistenz

Ein wichtiger Aspekt in der Erstellung einer Anwendung ist die lokale Persistenz beziehungsweise die Erhaltung der wichtigsten Daten. In Unity gibt es auf der einen Seite die Möglichkeit wichtige Daten über die sogenannten PlayerPrefs lokal zu speichern. Je nach Betriebssystem ist das ein anderer Ort auf dem Gerät, die Daten werden aber so auch nach dem Schließen der Anwendung erhalten. Dabei können hier Strings, Integer und Floats gespeichert und wieder ausgelesen werden, sodass es möglich ist beispielsweise Scores oder ähnliches einfach zu speichern. Da komplexere Daten nicht einfach zu speichern sind, wurden die Reisen mithilfe von JSON und dem Framework Json.net permanent erhalten. Da für jedes Betriebssystem hier ein unterschiedlicher Pfad des Geräts notwendig ist, wurde die finale Anwendung für Android entwickelt, kann aber mit Änderung des Pfades, wo die Datei gespeichert wird, auch für andere Betriebssysteme verwendet werden. Im Listing erkennt man die Grundstruktur einer mit Json.net gespeicherten Reise. Hier sieht man alle wichtigen Attribute, wobei hier nur ein Auszug aus der Packliste gezeigt wird und es sowohl keine Ausgaben wie auch

Verbindungen gibt. Wie man hier erkennen kann werden die Objekte in der Packliste mit Namen und einem boolean-Wert gespeichert, der anzeigt, ob man das Objekt schon eingepackt hat.

```
[
  {
    "Name": "Kroatien",
    "Date": "14.02.2019 - 18.02.2019",
    "PicturePath": "path,
    "PackingList": {
      "item": {
        "Kamm": false,
        "Impfpass": false,
        "Kamera": false,
        "Karte": false,
        "Kleid": false,
        "Rock": false,
        "Zahnpasta": false,
        "Zugticket/Bahncard/Flugticket": false
      }
    },
    "Budget": {
      "Total": 22,
      "Expenditures": {}
    },
    "Airplane": [],
    "Bus": [],
    "Train": []
  }
]
```

Listing 4.3: .json-Datei einer gespeicherten Reise ohne Verbindungen und Ausgaben

5

Realisierung

In diesem Kapitel geht es um die Realisierung der Anwendung. Folglich werden erst alle verwendeten Assets im Abschnitt 5.1 genauer beschrieben, dann werden die grundlegenden Entscheidungen, die für Unity getroffen werden mussten, erläutert. Schlussendlich werden die beiden Bildschirme, aus denen die Anwendung besteht, gezeigt und jede Funktionalität aufgeführt.

5.1 Verwendete Assets

In diesem Abschnitt werden alle verwendeten Assets aus dem Assetstore vorgestellt. Dazu gehören TextMeshPro Unterabschnitt 5.1.1, SSTools Unterabschnitt 5.1.2, AndroidNativePlugin 5.1.3 und JsonDotNet. JsonDotNet ist ein Framework, dass in Unity über den Assetstore angeboten wird. Da es ein Framework ist, wurde es in den Grundlagen Kapitel 2.3 schon erläutert und ist hier nicht noch einmal aufgeführt.

5.1.1 TextMeshPro

TextMeshPro ermöglicht es, verschiedenste Texte und Texteingabefelder individuell zu erstellen. So kann man hier eigene Schriftarten entwickeln und damit die Anwendung schöner gestalten, als es mit den bereits vorhandenen Texten möglich wäre. Außerdem können mit TextMeshPro die Tags, wie man sie aus HTML für die Schrift kennt, verwendet werden, was einen praktischen und schnellen Umgang ermöglicht [13]. Ein Beispiel hierfür sieht man in Abbildung 5.1. Hier wurde der Text „EXAMPLE OF REVEALING TEXT ONE CHARACTER AT A TIME IN TEXTMESHPRO“ gestaltet und mithilfe von

Tags farblich unterschieden. Dabei sieht man im linken Feld den modifizierten Text und auf der rechten Seite alle änderbaren Eigenschaften.

Abbildung 5.1: Beispiel eines TextMeshPro Textes, der mit Tags gestaltet wurde [14]

5.1.2 SSTools

SSTools oder wie im Assetstore beschrieben SS-System 3.0: UI and Scene Manager, stellt verschiedenste Methoden bereit, mit denen man das User Interface leichter gestalten und kontrollieren kann. Die meisten Funktionen wie das Versenden von Daten zwischen verschiedenen Szenen oder das Steuern von Sound wurden nicht verwendet. Die Hauptfunktionalität bestand darin, für Unity Benachrichtigungen anzuzeigen, die in Android Studio mithilfe von Toasts verwirklicht wurden [15]. So sieht man im Beispiel die Anzeige eines Toast, wenn ein Eingabefeld noch nicht ausgefüllt wurde.

```
1 if (string.IsNullOrEmpty(MName.text))
2         {
3             SSTools.ShowMessage("Please enter name",
4             SSTools.Position.bottom, SSTools.Time.twoSecond);
5         }
```

Listing 5.1: Nutzung von SSTools als Toast

5.1.3 AndroidNativePlugin

Das Android Native Plugin in der frei verfügbaren Version enthält die Funktionen zum Öffnen eines Bildes in Android, zum Erstellen eines Android Spinner oder zum Anzeigen einer Dialogbox. Verwendet wurde hier das Öffnen eines Bildes, da dies in Unity nicht so einfach zu realisieren ist wie beispielsweise in Android Studio und deswegen als Asset eingebaut werden kann [16]. Darüber hinaus wurden auch die Dialogboxen verwendet, um sicherzustellen, dass der Nutzer auch wirklich diese Entscheidung treffen will.

5.2 Grundlegende Entscheidungen für Unity

In diesem Abschnitt sollen alle Entscheidungen und Konzepte, die speziell für das Programm Unity getroffen wurden, vorgestellt werden. So werden zunächst die Konzepte ScrollView, Anchor und Scenes und Gameobjects allgemein erläutert, um danach auf die Verwendung in dieser Anwendung einzugehen.

5.2.1 ScrollView

Ein grundlegendes Konzept, das für diese Anwendung gebraucht wurde, ist die ScrollView. Diese gibt es nicht bereits fertig, sondern muss selbst entworfen werden. In dieser Anwendung besteht die ScrollView immer aus einem Scroll Rect, welches es so schon für Unity gibt, einem Viewport, der dann den Inhalt enthält, und einer Scrollbar. Das Scroll Rect enthält die Funktionen zum Scrollen mit dem Finger und muss nicht selbst implementiert werden. Der Viewport ist entscheidend, da er den eigentlichen Inhalt enthält. Dieser muss als Vertical Layout Group definiert werden, da sonst die Kindelemente nicht untereinander, mit dem gleichen Abstand angeordnet werden. Des Weiteren muss der Inhalt ebenfalls das Script Content Size Fitter beinhalten, da dieses dafür sorgt, dass alle Elemente die gleiche, angepasste Größe haben. Wichtig ist nun noch, dass man der ScrollView Elemente dynamisch hinzufügen kann. Dazu muss ein weiteres Script geschrieben werden, dass dann einzelne Elemente mit dem richtigen Inhalt erstellt und dieser ScrollView hinzufügt. Dazu müssen im voraus Prefabs erstellt werden. Diese sind

Vorlagen für Elemente und werden dann bei Gebrauch instantiiert. Da jede ScrollView ein bisschen anders aufgebaut ist, gibt es für jede ein eigenes Script und dazu passende Prefabs. Die Srcollbar als letztes Element bezeichnet die beiden Leisten, mit denen man auch den Inhalt verschieben kann, wobei für diese Anwendung die horizontale Scrollbar deaktiviert wurde.

5.2.2 Anchor

In Unity gibt es für die Gestaltung der Seite für verschiedene Bildschirmformate, sogenannte Anchors. Mit diesen kann man die Position des Objekts ziemlich genau festlegen, indem man diese an Positionen verschiebt, die man möchte. Bei der Übertragung auf ein bestimmtes Format werden die positionierten Anchors beibehalten und das Design bleibt grundlegend das selbe. Insgesamt gibt es vier solcher Anchors für jedes Objekt und diese können left, center, right und top, middle, bottom oder individuell ausgerichtet werden. In diesem Fall wurden die Anchor individuell positioniert, da es so möglich ist ein genaueres Design zu verwirklichen.

5.2.3 Scenes und Gameobjects

Die Einteilung in Unity erfolgt entweder in Scenes oder über aktive und inaktive Gameobjects. Die wichtigsten Bildschirme, der Hauptbildschirm und der Übersichtsbildschirm, wurden als eine eigene Scene entwickelt. Diese werden für essentielle Teile, im Normalfall Levels, verwendet. Darüber hinaus gibt es die Möglichkeit einzelne Anzeigen über Gameobjects zu definieren und diese dann aktiv zu setzen, wenn sie gebraucht werden. Dies geschieht über sogenannte Event Trigger, die man direkt in Unity einem Gameobject hinzufügen kann. Anschließend wählt man ein Event, bei dem die Aktion ausgeführt werden soll und das gewünschte Element oder Skript. Alle anderen Anzeigen, wie eine neue Reise erstellen oder eine neue Verbindung hinzufügen, wurden als Gameobjects definiert und bei Gebrauch aktiviert.

5.3 Hauptbildschirm

In diesem Abschnitt wird der Hauptbildschirm, welcher gleich nach Start der Anwendung erscheint, näher erläutert. Dazu wird zunächst der Grundaufbau skizziert, um dann die Hauptfunktion, das Erstellen einer neuen Reise, zu erklären. Der Hauptbildschirm ist eine der beiden Szenen, die in Unity erstellt wurden und mit dem der Grundaufbau der Anwendung definiert wird. Diese hat den Index 0 und wird deswegen gleich am Anfang aktiviert.

5.3.1 Übersicht

Die Übersicht der Reisen ist die wichtigste Anzeige, da sie direkt nach Starten der Anwendung angezeigt wird. In Abbildung 5.2 erkennt man den Event Trigger, der dem Pluszeichen hinzugefügt wurde.

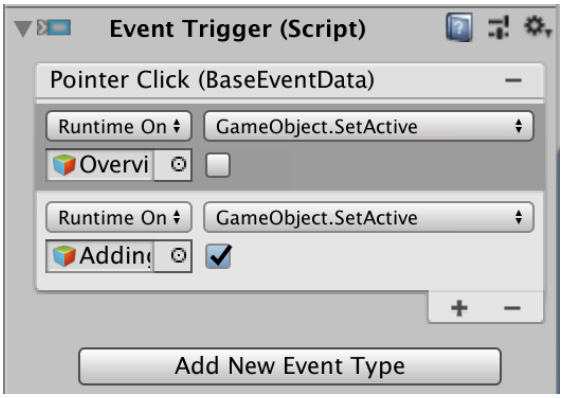

Abbildung 5.2: Event Trigger (Script) das einem einzelnen Element hinzugefügt wurde

Aufgerufen wird dieser bei einem Pointer Click Event, also wenn das Zeichen angeklickt wird. Anschließend wird die Übersicht (hier Overview) inaktiv und die Seite zum Hinzufügen einer Reise (hier AddingJourney) aktiv gesetzt. Die Übersichtsanzeige ist in dieser Szene zu Beginn aktiv und wird deaktiviert, wenn ein Element auf dieser Seite ausgewählt wird. In Abbildung 5.3 erkennt man genau diesen Bildschirm. Der Bildschirm ist vom Design ähnlich zu dem in den Mock-ups gezeigten Entwurf. Jedoch erkennt man,

das zwar das Grundkonzept gleich geblieben ist, sich aber bei der Gestaltung der einzelnen Reiseelemente etwas geändert hat. So besitzt jede Reise nun ihr individuelles Bild, das man festlegen kann. Darüber hinaus hat jedes Reiseelement einen Namen und ein Datum mit Anfangs- und Endtag. Diese Elemente befinden sich alle in einer ScrollView und diese wird beim Hinzufügen neuer Elemente dynamisch erweitert. Außerdem kann man bei sehr vielen Elementen mit dem Finger nach oben oder unten scrollen. Des Weiteren verfügt jedes der Elemente über einen Listener, der erkennt ob ein Element ausgewählt wurde und für dieses die Übersicht über die Verbindungen usw. öffnet (siehe Abschnitt 5.4).

```
1 public void OnPointerClick(PointerEventData eventData)
2     {
3         TextMeshProUGUI text = GetComponentInChildren
4         <TextMeshProUGUI>();
5         PlayerPrefs.SetString("journeyclicked", text.text);
6         SceneManager.LoadScene(SceneManager.
7         GetActiveScene().buildIndex + 1);
8     }
```

Listing 5.2: Listener der Reiseelemente

Im unteren Bereich erkennt man ein Plus-Zeichen mit zugehöriger Schrift „Add new Journey “. Mithilfe von diesem kann man eine neue Reise hinzufügen, was in Unterabschnitt 5.3.2 näher erläutert wird.

5.3.2 Neue Reise erstellen

Nachdem man auf das Plus oder die dazugehörige Schrift geklickt hat, öffnet sich der Bildschirm in Abbildung 5.4. Bei diesem Vorgang wird die Übersicht über die Reisen inaktiv gesetzt und im Gegenzug die Ansicht für die neuen Reisen aktiviert.

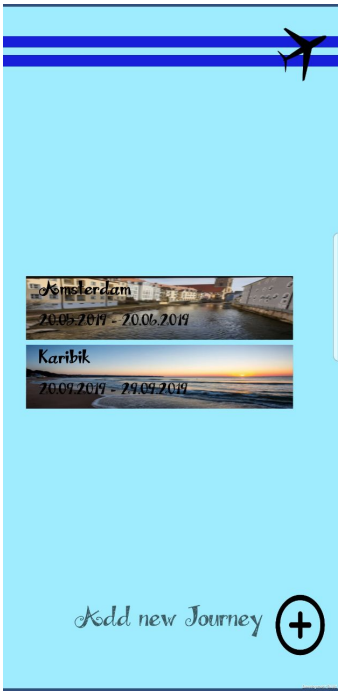

Abbildung 5.3: Die Übersichtsseite der Reisen, hier mit einer Reise nach Amsterdam und einer Reise in die Karibik

Das obere, weiße Viereck dient dazu, ein Bild anzuzeigen, nachdem man es ausgewählt hat. Um dies zu erreichen, betätigt man den Button „Choose Image “ und wählt dann ein Bild aus seiner Galerie aus. Damit eine neue Reise hinzugefügt werden kann, müssen alle Eingabefelder ausgefüllt werden. Wie die Hints in den Eingabefeldern verraten sind diese für den Namen, das Datum und das Budget gedacht. Lässt man eines dieser Felder leer oder gibt für das Budget keine Zahl ein, erscheint ein Toast mit dem passenden Hinweis. Hat man nun alle Felder ausgefüllt, kann man mithilfe des Create-Buttons eine neue Reise erstellen, die dann unten in die Liste eingefügt und in die Json-Datei gespeichert wird. Möchte man doch keine Reise erstellen, kann man das Ganze mit dem Cancel-Button abbrechen. Dabei wird dann diese Ansicht bzw. das passende Gameobject dafür deaktiviert und wieder das der Übersicht angezeigt.

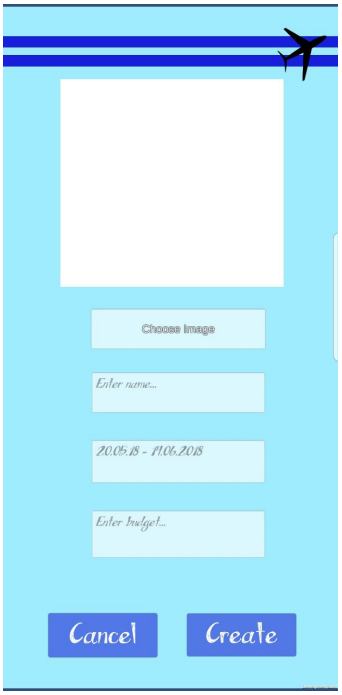

Abbildung 5.4: Bildschirm zum Hinzufügen einer Reise mit Bild und drei Eingabefeldern

5.4 Übersichtsbildschirm

Der Übersichtsbildschirm wird nach der Auswahl einer Reise aufgerufen und zeigt zu Beginn eine Übersicht der Verbindungen an, die am Anfang noch leer ist. Dieser Bildschirm ist die zweite wichtige Szene, die es in der Anwendung gibt. Der Grundaufbau ist dabei so, dass in der oberen linken Ecke immer ein Zurück-Pfeil zu sehen ist, mit dem man wieder zum Hauptbildschirm und damit der Reiseübersicht gelangen kann. Außerdem ist daneben der Name der ausgewählten Reise und ein Mülleimer Symbol angeordnet. Mithilfe des Mülleimers lässt sich eine Reise löschen, worauf in Unterabschnitt 5.4.8 noch genauer eingegangen wird. Darunter sieht man die Begriffe Connection, Budget und Packing List, welche jeweils einen Tab darstellen, den man auswählen kann. Zu Beginn ist immer der Connection-Tab aktiv, wählt man im Folgenden einen anderen Tab, wird dieser farblich markiert, damit man sofort sieht, wo man sich befindet. Die Tabs wurden mithilfe von drei verschiedenen Buttons realisiert, die je nach Zustand

eine andere Farbe annehmen. Damit auch jeweils eine neue Ansicht gezeigt wird, kann man in Unity für Buttons bei On Click () eine Funktion RectTransform.SetAsLastSibling aufrufen und das dafür gewählte Objekt wird in den Vordergrund gezeichnet.

5.4.1 Übersicht über Verbindungen

In dem Tab Connection erhält man eine Übersicht über die bereits bestehenden Verbindungen, wie man in Abbildung 5.5 erkennt.

Abbildung 5.5: Bildschirm über die gesamten Verbindungen der Reise. Hier Hinreise mit dem Bus und Heimreise mit dem Flugzeug

Hier sieht man eine Bus- und eine Flugverbindung. Die Art der Verbindung erkennt man sowohl an dem Symbol am linken Rand des Verbindungselements, als auch an der jeweiligen Farbe. So sind alle Busse in der Farbe Grün, Flüge in der Farbe Rot und Züge in der Farbe Lila kodiert. In jedem Verbindungselement steht der Abfahrts- und Ankunftsort, das Datum und die Uhrzeit. Auch hier besitzt jede Verbindung einen Listener, damit bei einem Klick auf eine Verbindung die Details angezeigt werden (siehe Unterabschnitt

5.4.3). Außerdem sind sie wie die Reiseelemente in einer ScrollView angeordnet, damit man einen schnellen Überblick über alle Verbindungen erlangen kann. Am unteren Rand des Bildschirms kann man in blauer Schrift und unterstrichen den Text „Show route on map “ lesen, der deswegen so gestaltet ist, da es sich um einen Link handelt. Klickt man diesen an, gelangt man zu Google Maps, bei dem dann die gesamte Route und Alternativverbindungen angezeigt werden. Da es in Unity keine direkte Methode gibt die Anwendung Google Maps zu öffnen, wird hier Google Maps über die passende URL aufgerufen. Dazu wurde die Methode Application.OpenURL mit dem string ("https://www.google.com/maps/dir/?api=1&origin=1&destination=0&travelmode=transit", destination, origin) ausgeführt. Dieser setzt sich aus der allgemeinen Url für Google Maps, dem Start (origin), dem Ziel (destination) und der Art zu Reisen (travelmode) zusammen. Als Start wird dabei die zuerst hinzugefügte Verbindung, als Ziel die zuletzt hinzugefügte und als travelmode transit, also öffentliche Verkehrsmittel, gewählt. Als letztes gibt es auf diesem Bildschirm ein weiteres Plus-Zeichen und die dazugehörige Schrift „Add new connection... “, mit dem man auch hier eine neue Verbindung hinzufügen kann. Genaueres dazu wird im nächsten Unterabschnitt 5.4.2 erklärt.

5.4.2 Neue Verbindung hinzufügen

Um eine neue Verbindung hinzuzufügen, muss man auf das Plus im Connection-Tab klicken. Anschließend öffnet sich zunächst der in Abbildung 5.6 gezeigte Bildschirm. Dieser besitzt auch in der linken, oberen Ecke einen Zurück-Pfeil, mithilfe dem man zurück zum Übersichtsbildschirm gelangen kann. Außerdem wird durch die Frage „How do you want to travel? “ darauf aufmerksam gemacht, dass der Nutzer sich hier eine Art der Verbindung aussuchen soll. Dies kann man durch den Klick auf eines der drei Bilder bestätigen. Zur Auswahl stehen hierbei eine Zug-, Bus-, oder Flugverbindung, je nachdem wie man verreisen möchte. Unter jedem Bild wird zur Vorbeugung von Verwechslungen die Art der Verbindung nochmals aufgeschrieben. Hat man nun eine Verbindungsart gewählt, gelangt man zum Bildschirm in Abbildung 5.7.

In diesem Fall wurde die Verbindung Flug gewählt und dementsprechend erscheint in der linken, oberen Ecke das Flugzeugsymbol. Für die anderen Verbindungen erscheint

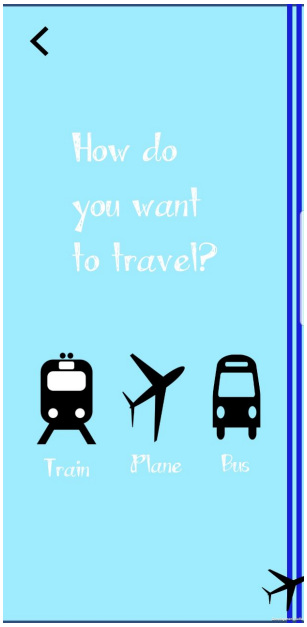

Abbildung 5.6: Bildschirm zur Auswahl der Verbindungsart

Abbildung 5.7: Bildschirm, bei dem man die Angaben zur Verbindung macht (hier: Flug)

jeweils deren Symbol, welches schon im vorherigen Bildschirm gezeigt wurde. Auch hier hat man die Möglichkeit über den Zurück-Pfeil in der linken, oberen Ecke einen Schritt zurück zu gehen und eine neue Verbindungsart auszuwählen. Damit man weiß, dass es nun darum geht, weitere Informationen einzufügen, wird dies gleich zu Beginn mit der Zeile „Adding Information... “ angezeigt. Je nach Verbindungsart gibt es unterschiedliche Eingabefelder die ausgefüllt werden müssen. Die Hints in den Felder dienen zur Hilfe, welche Informationen verlangt werden. Hat man alle Felder ausgefüllt, kann man die Verbindung über den Okay-Button hinzufügen. Diese Verbindung wird über eine spezielle Methode in der Klasse SaveToJSON der .json Datei an der richtigen Stelle hinzugefügt. Sind noch Eingabefelder leer geblieben, so erhält man auch hier über einen Toast den Hinweis, diese auch noch auszufüllen. Will man doch keine Verbindung hinzufügen, gibt es die Möglichkeit, über den Cancel-Button wieder zur Übersichtsseite zu gelangen.

5.4.3 Details der Verbindung anzeigen

Möchte man noch nähere Details zu seiner Verbindung, wie beispielsweise das Gate haben, kann man diese über einen Klick auf die jeweilige Verbindung erlangen.

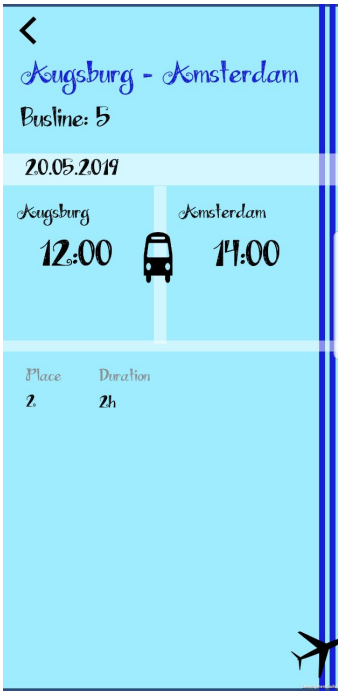

Abbildung 5.8: Bildschirm für die Details der Verbindung

In Abbildung 5.8 sieht man den Bildschirm der Detailansicht. Hier sind alle vorher eingegebenen Informationen noch einmal übersichtlich dargestellt. Auch hier gelangt man mit dem Zurück-Pfeil wieder zur Übersichtsseite.

5.4.4 Übersicht über Budget

Der zweite Tab der Übersichtsseite enthält eine Übersicht über das Budget. In Abbildung 5.9 sieht man, dass einen Großteil der Seite ein Kreisdiagramm einnimmt.

In Grün wird hierbei das noch verfügbare Geld gezeigt, in diesem Fall hat man noch sein ganzes Budget zur Verfügung. In Rot werden die Ausgaben prozentual zum Gesamtbudget angezeigt und das Kreisdiagramm soweit gefüllt. Diese Farben wurden so gewählt, da

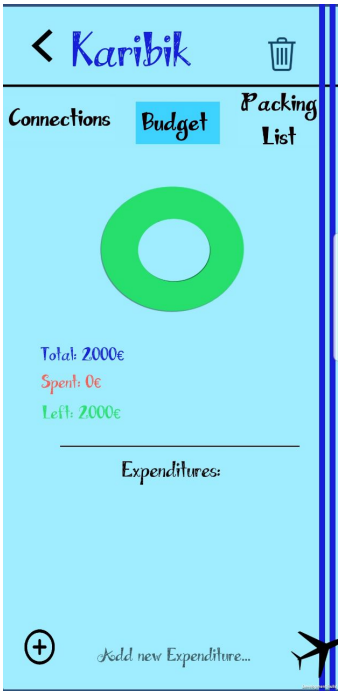

Abbildung 5.9: Bildschirm zur Übersicht des Budgets und den Ausgaben

sie üblicherweise im Kontext von Geld so genutzt werden und damit schon in Verbindung gebracht werden. Passend zu den Farben im Kreisdiagramm wurden auch die Farben der darunterliegenden Schrift, die den genauen Betrag des noch übrigen und schon verbrauchten Budgets zeigt, gestaltet. Das Gesamtbudget wurde in der recht neutralen Farbe Blau, passend zum allgemeinen Design, gefärbt. Zur Budgetübersicht gehören des weiteren auch die verschiedenen Ausgaben, die mit einem Strich abgetrennt unter dem Kreisdiagramm stehen. Diese werden auch in einer ScrollView angeordnet und ein Ausgabenelement besteht dabei aus dem Namen und dem angegebenen Betrag, der dann in der oberen Ansicht verrechnet wird. Eine weitere Ausgabe kann man hier auch über das Plus-Zeichen und der dazugehörigen Schrift „Add new Expenditure... “ hinzufügen. Geöffnet wird hierbei der Bildschirm der in Unterabschnitt 5.4.5 gezeigt wird.

5.4.5 Neue Ausgabe hinzufügen

Durch Klicken auf das Plus-Zeichen im Tab Budget öffnet sich der Bildschirm, den man in Abbildung 5.10 sehen kann.

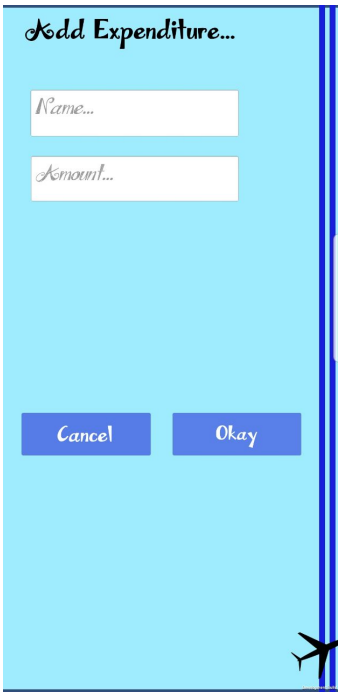

Abbildung 5.10: Bildschirm zum Hinzufügen einer Ausgabe mit Name und Betrag

Dieser ist eher schlicht gehalten und enthält nur die Hauptfunktionen. So besteht er aus der Überschrift Add Expenditure..., zwei Eingabefeldern für den Name und den Betrag und zwei Buttons. Der Cancel-Button beendet hierbei das Hinzufügen einer Ausgabe und man gelangt wieder zum Tab Budget. Der Okay-Button schließt hingegen das Hinzufügen der Ausgabe ab, diese ist nun in der Budgetübersicht wiederzufinden und das Kreisdiagramm wurde angepasst.

5.4.6 Übersicht über Packliste

Der letzte der drei Tabs PackingList, enthält die Übersicht der Packliste. In Abbildung 5.11 sieht man das nun alle Elemente der Packliste untereinander angeordnet worden

sind und man durch diese scrollen kann. Außerdem wurden die einzelnen Elemente nun als Checkbox dargestellt, sodass man diese einfach abhaken kann. Sie sind darüber hinaus alphabetisch gegliedert, um dem Nutzer eine schnellere Suche zu ermöglichen. In der Abbildung wurden beispielsweise die Elemente Kulturtasche und Liege schon eingepackt. Auch hier besteht die Möglichkeit neue Elemente über das Plus-Zeichen hinzuzufügen, falls man noch nicht alle gewollten Elemente in der Packliste hat.

Abbildung 5.11: Bildschirm mit der Packliste der Reise mit Checkboxen zum Abhaken

5.4.7 Neues Objekt hinzufügen

Um ein neues Objekt der Packliste hinzuzufügen, wurde im Tab PackingList das Plus-Zeichen angeklickt. Daraufhin öffnet sich der Bildschirm von Abbildung 5.12, der wie der Bildschirm zum Hinzufügen neuer Ausgaben, sehr schlicht designend ist. So besteht dieser auch nur aus einer Überschrift und einem einzigen Eingabefeld für den Namen. Anschließend wird das Element bei Bestätigung auf den Okay-Button der Liste hinzuge-

fügt und kann dann abgehakt werden. Möchte man doch kein Element erstellen, kann man dies mit einem Klick auf den Cancel-Button verhindern.

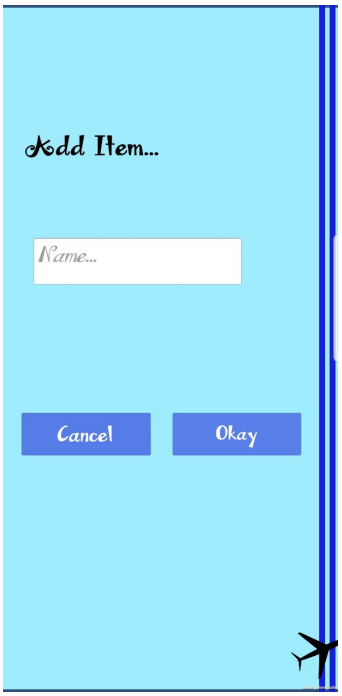

Abbildung 5.12: Bildschirm zum Hinzufügen eines Objektes zur Packliste

5.4.8 Reise löschen

Die letzte Funktion der Anwendung ist das Löschen einer Reise. Ist die Reise bereits vergangen oder wird doch nicht angetreten, kann man die Reise auch wieder löschen. Nach dem Auswählen der zu löschenden Reise entfernt man diese durch Klicken auf das Mülleiner-Symbol in der rechten, oberen Ecke. Danach erscheint ein Dialogfenster, in dem man gefragt wird, ob man die Reise wirklich löschen möchte. In Abbildung 5.13 kann man dieses Fenster erkennen.

Man hat nun zwei Möglichkeiten weiter vorzugehen. Wählt man Yes wird die bestehende Reise vollständig gelöscht und man gelangt wieder zum Hauptbildschirm und damit zur Übersicht über die verbleibenden Reisen. Die zweite Möglichkeit besteht darin, No zu

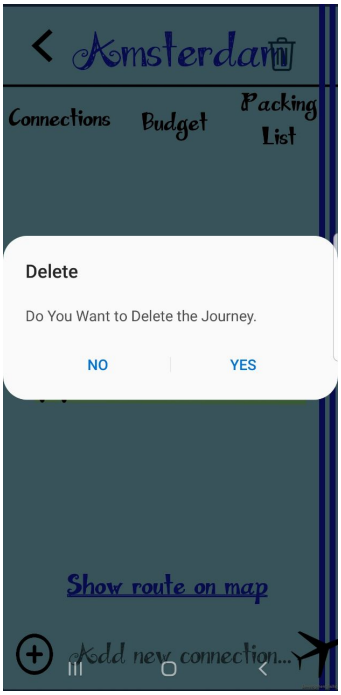

Abbildung 5.13: Bestätigungsdialog zum Löschen einer Reise

wählen und die Reise bleibt somit bestehen. Der Dialog schließt sich nach dieser Wahl und man bleibt weiterhin im Übersichtsbildschirm.

6

Anforderungsvergleich

Im Anforderungsvergleich soll festgestellt werden, welche Anforderungen aus der Anforderungsanalyse realisiert, geändert oder weggelassen wurden. Im Folgenden werden zunächst die nicht-funktionalen Anforderungen mit der Realisierung verglichen, um danach auf die funktionalen Anforderungen einzugehen.

6.1 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen stehen auch für die Qualität der Anwendung und wurden deswegen vollständig in der Anwendung umgesetzt. Die Dokumentation wurde wie in C# üblich über Kommentare im XML-Stil erstellt, die man sich dann generieren lassen kann.

```
1     /// <summary>
2     /// The method checks if the fields are empty and if not
3     /// generates a new itemmodel with the values from
4     /// the fields. Also saves an expenditure in the .json-File
5     /// </summary>
6     public void UpdateItem(){
7     ...
8     }
```

Listing 6.1: Beispiel Kommentar für die Dokumentation

Die nicht-funktionale Anforderung Verfügbarkeit wurde über verschiedene if-cases abgedeckt, sodass auf Benutzereingaben korrekt reagiert wird. Q03, das Zeitverhalten, wird durch die hohe Performance von Unity erreicht. Auch die Robustheit wurde ausreichend getestet und die Anwendung wurde in 100 Fällen nicht geschlossen. Zuletzt wird die leichte Bedienbarkeit durch bekannte Symbole und dazu erklärender Schrift in der englischen Sprache erzielt.

6.2 Funktionale Anforderungen

Bei den funktionalen Anforderungen wurden die Anforderungen FA01 - FA07 vollständig erfüllt, da sie den Grundanforderungen der Anwendung entsprechen. Darüber hinaus sind auch FA10 - FA13 über die Verbindungen genau so realisiert worden, FA15 - FA18 entsprechen den Funktionen für die Bildschirme, die ebenfalls erfüllt wurden. Als letztes wurde FA21, die Funktion eine Reise zu löschen, in die Anwendung eingebaut.
Geändert wurde FA09, da FA08, die Kategorien einer Packliste, entfernt wurde und damit auch das Item einer Packliste keine Kategorie mehr besitzt. Dies wurde so entschieden, da es in Unity nicht einfach ist dafür eine geeignete Ansicht zu erstellen, bei der man beispielsweise die Kategorien ausklappen könnte. Dafür existieren auch keine frei verfügbaren Assets, weswegen FA08 weggelassen wurde.
Ein paar Anforderungen wurden aus verschiedenen Gründen nicht in die Anwendung mit aufgenommen. So ist FA14, die Funktion weitere Verbindungen anzuzeigen, schwer umzusetzen, da die API der deutschen Bahn nicht so leicht abzurufen ist und selbst dann nur für Fernzüge zur Verfügung steht. Dies würde die Anwendung nicht in der Form unterstützen, die gefordert wäre und wurde aufgrund des zu hohen Aufwands nicht realisiert. Die fehlenden Anforderungen beinhalten die Möglichkeit verschiedenste Dinge in der Anwendung zu löschen, die aber nicht hoch priorisiert waren und da man eine Reise vollständig löschen kann, nicht umgesetzt wurden. Im Gesamten wurden die wichtigsten und grundlegenden Anforderungen realisiert, die auch mit einem + oder ++ versehen waren. Weggelassen wurden nur die Anforderungen, die im voraus auch schon als nicht zwingend notwendig erachtet wurden oder nicht umsetzbar waren.

7

Verwandte Arbeiten

In diesem Kapitel geht es darum, sich einen Überblick darüber zu verschaffen, was es schon an ähnlichen Arbeiten gibt. Im Folgenden werden zwei Anwendungen vorgestellt, die man im AppStore kostenlos erhalten kann und die sich ebenfalls mit dem Thema Reiseplanung beschäftigen. Die erste Anwendung ist dabei Google Trips, die in Abschnitt 7.1 gezeigt und deren Funktionalitäten erläutert werden. Die nächste Anwendung ist TripIt, die dann in Abschnitt 7.2 im Detail untersucht wird.

7.1 Google Trips

Google Trips ist, wie die in der Arbeit vorgestellte Arbeit, eine Anwendung zur Reiseplanung. Allerdings spezialisiert sich diese Anwendung auf andere Anwendungsgebiete und enthält keine Informationen zum Budget oder der Packliste. In Google Trips werden auf der Startseite auch die gesamten Reisen angezeigt. Anders als bei dieser vorgestellten Anwendung werden in Google Trips bereits passende Bilder ausgewählt, die man sich aber nicht selber aussuchen kann. In Abbildung 7.1 erkennt man den Bildschirm, nachdem eine bestimmte Reise ausgewählt wurde. Hier sieht man dann die Möglichkeiten, die diese Anwendung bietet. Hier ist es möglich seine Reservierungen einzutragen, dazu zählen sowohl Verbindungen als auch Hotelbuchungen. Darüber hinaus kann an diesem Ort nach weiteren Aktivitäten gesucht, bestimmte Orte gespeichert werden, die man besuchen möchte, Pläne für den Tag erstellt oder nach Restaurants in der Umgebung gesehen werden[17].

Allerdings kann man nicht immer auf die Pläne für den Tag zugreifen, da dies nur möglich ist, wenn die Anwendung Sehenswürdigkeiten in der Nähe erkennt. Genauso

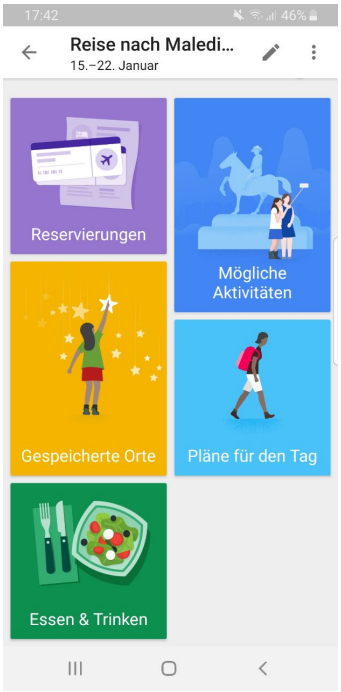

Abbildung 7.1: Bildschirm nach Auswahl einer bestimmten Reise

verhält es sich mit dem Button Essen & Trinken, der nur verfügbar ist, wenn es auch Restaurants gibt. Außerdem erschließen sich die Funktionen der gespeicherten Orte nicht, da dort keine angezeigt werden und man nicht darauf hingewiesen wird, wie man welche hinzufügt. Als letztes sollte man anmerken, dass zwar die Hauptseiten schön gestaltet sind, will man dann aber eine Reise oder Verbindung hinzufügen, sind nur die einfachsten Schaltflächen vorhanden und es wurde komplett auf Farbe verzichtet.

7.2 TripIt

TripIt ist eine weitere Anwendung zur Planung einer Reise. TripIt war zunächst ein kostenloser online Reiseplaner, mithilfe dessen man seine Reisen verwalten konnte und es passend für jede Reise einen Reiseguide, Wettervorhersagen oder Aktivitäten gab, die man eintragen konnte [18]. Mittlerweile gibt es von TripIt eine mobile Anwendung, mit der man seine Reisen verwalten kann [19]. In Abbildung 7.2 erkennt man hierbei

den Startbildschirm der Anwendung. Dabei sind die Reisen in noch kommende und schon vergangene Reisen eingeteilt und über den Plus-Button können weiter hinzugefügt werden. Außerdem kann man in der unteren Leiste zwischen seinen Reisen, Benachrichtigungen, seinem Profil, „Unfiled “ und weiteren Einstellungen wechseln. In der nächsten Abbildung 7.3 ist nun eine spezielle Reise zu sehen. Über das Plus können hier auch Reservierungen hinzugefügt werden. Das Design hierbei ist wie der Übersichtsbildschirm sehr schlicht gehalten und enthält sehr wenig farbige Elemente.

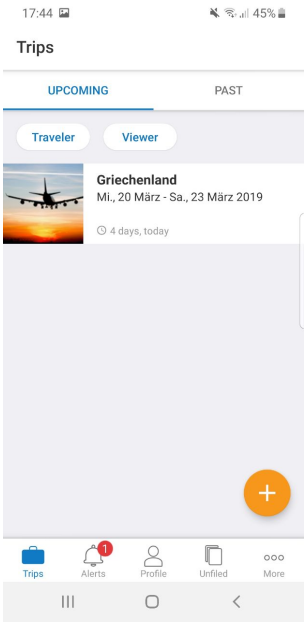

Abbildung 7.2: Startbildschirm der Anwendung

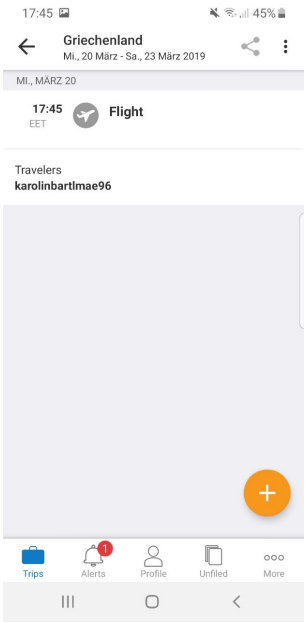

Abbildung 7.3: Bildschirm nach Auswahl einer bestimmten Reise

Weitere Funktionen sind in dieser Anwendung nicht enthalten. Folglich wird hier der Fokus allein auf die Reise an sich ohne weitere Informationen außer den Reservierungen gelegt.

8

Zusammenfassung und Ausblick

Zusammenfassend kann man sagen, dass in dieser Arbeit eine Anwendung entstanden ist, die zur Reiseplanung beiträgt und auch für diese verwendet werden kann. Die Anwendung stellt Hilfen für das Packen, das Planen vom Budget und die Verbindungen zur Verfügung. Allerdings sollten hier auch einige Punkte genannt werden, die die Erstellung einer Businessanwendung mit Unity erschweren. Als erster Aspekt ist Unity grundsätzlich nicht dafür ausgelegt solche Anwendungen zu realisieren, weswegen einige grundsätzliche Funktionen noch in der Entwicklung sind. Hauptproblem sind dabei Toasts, Dialogfenster, Scrollviews und viele weitere, die noch nicht integriert sind und im Assetstore erworben oder selbst aufwendig implementiert werden müssen. Außerdem kann auf ein bestimmtes Betriebssystem wie Android nicht leicht zugegriffen werden wie das in Android Studio mit Intents bereits möglich ist. Trotz dieser Nachteile gibt es dennoch genügend positive Aspekte, die diese wieder ausgleichen. Als erstes ist Unity ein Programm, das, wenn man es verstanden hat, sehr einfach und schnell zu bedienen ist und vor allem die Navigation zwischen unterschiedlichen Szenen oder aktiven Gameobjects ist positiv hervorzuheben. Darüber hinaus sind auch die EventTrigger sehr effektiv, direkt im Programm aktivierbar und müssen nicht implementiert werden. Auch die Speicherung der Daten über die PlayerPrefs ist eine effektive Methode, die einem die Intents in Android Studio erspart. Der Absturz und Neustart der Anwendung ist damit ebenfalls kein Problem mehr, da die Daten persistent gespeichert sind.
Folglich kann das Programm zur Erstellung einer Anwendung, die nicht einem Spiel entspricht, gut verwendet werden und hat wegen seinem Aufbau einige Vorteile, sodass es sich lohnen kann, Unity einem anderen Programm vorzuziehen.

Für die Zukunft könnte man sich einige Erweiterungen der Anwendung vorstellen, die die Verwendung noch einfacher und sinnvoller machen würde. So konnte die API der deutschen Bahn nicht eingebaut werden, da diese noch keine vollständigen Informationen enthält. Es ist durchaus vorstellbar, dass diese Informationen in nächster Zeit auch für alle Entwickler frei erhältlich ist. Darüber hinaus kann man die Anwendung im Vergleich zu den verwandten Arbeiten auch bezüglich Hotelreservierungen oder geplanten Ausflügen erweitern. Zur Übersichtlichkeit könnten die Reisen auch in vergangene und noch ausstehende eingeteilt werden, damit man die relevanten Reisen zuerst sieht. Weiterhin werden immer mehr Funktionen für Unity implementiert und sind im Assetstore frei erhältlich, sodass es bald mehrere Möglichkeiten zur sinnvollen Darstellung geben wird, als momentan verfügbar sind. Insgesamt gibt es einige Punkte die man sich im Ausblick in die Zukunft vorstellen kann, noch zu bearbeiten, um eine optimale Anwendung zu erhalten.

Literaturverzeichnis

[1] Anika, R.: DIE MOBILE ZUKUNFT IN ZAHLEN: STATISTA VERÖFFENTLICHT REPORT ZU DEN MEGATRENDS UNSERER ZEIT. https://mobilbranche.de/2017/06/die-zukunft-zahlen (2017) [Online; accessed 14-03-2019].

[2] Lara, H.: VIR Daten & Fakten zum Online-Reisemarkt. Verband Internet Reisevertrieb e.V. (VIR) (2019)

[3] Technologies, U.: The world's leading real-time creation platform. https://unity3d.com/de/unity (2019) [Online; accessed 14-03-2019].

[4] Crockford: The application/json Media Type for JavaScript Object Notation (JSON). The Internet Society (2006)

[5] Newtonsoft: Json.net. https://www.newtonsoft.com/json (2019) [Online; accessed 14-03-2019].

[6] Holger Wagner, O.A.: Das chronische Problem der Anforderungsanalyse und die Frage Fehler vermeiden oder früh entdecken? Lehrstuhl für Wirtschaftsinformatik, Systementwicklung (2005)

[7] Braun, M.: Nicht-funktionale Anforderungen. Lehrstuhl für Programmierung und Softwaretechnik (2016)

[8] Ebert, C.: Systematisches Requirements Engineering. dpunkt.verlag (2019)

[9] Klaus, M.: Business Intelligence (Teil 3): Datenmodellierung ? Relationale und Multidimensionale Modelle). https://www.tecchannel.de/a/business-intelligence-teil-3-datenmodellierung-relationale-und-multidimensionale-modelle,1744994,3 (2008) [Online; accessed 18-03-2019].

[10] Wikipedia: Model-view-controller). https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller (2019) [Online; accessed 18-03-2019].

[11] Glenn E. Krasner, S.T.P.: A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. ParcPlace Systems (1998)

[12] ninjamock.com: Slice your Wireframe work in half). https://ninjamock.com/ (2019) [Online; accessed 19-03-2019].

[13] Studios, D.N.: Online Documentation. http://digitalnativestudios.com/textmeshpro/docs/ (2019) [Online; accessed 23-03-2019].

[14] Zolran: Text Mesh Pro - Text Reveal Effect. https://www.youtube.com/watch?v=U85gbZY6oo8 (2015) [Online; accessed 23-03-2019].

[15] Pham, A.: SS-System 3.0: UI and Scene Manager. https://assetstore.unity.com/packages/tools/ss-system-3-0-ui-and-scene-manager-12947 (2016) [Online; accessed 23-03-2019].

[16] Hashmi, W.: Android Native Plugin Free. https://assetstore.unity.com/packages/tools/integration/android-native-plugin-free-103738 (2018) [Online; accessed 23-03-2019].

[17] google: GoogleTrips. https://get.google.com/trips/ (2019) [Online; accessed 20-03-2019].

[18] Technologies, C.: Tripit. https://www.tripit.com/web (2019) [Online; accessed 20-03-2019].

[19] Lynda M. Applegate, Gabriele Piccoli, K.B.: TripIt: The Traveler's Agent. Harvard Business School (2008)

Abbildungsverzeichnis

Name: Karolin Bartlmä Matrikelnummer: 909282

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Karolin Bartlmä