**Casino App**

**Fullstack Angular Applikation**

**Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte Beschreibung**

Individuelle praktische Arbeit

Paul Odiana Pietzko

Kantonsschule Sargans

Exp. Christian Steiner

05.03.2024

# Inhalt

[Inhalt 1](#_Toc165378980)

[Kurzfassung 5](#_Toc165378981)

[Ausgganssituation 5](#_Toc165378982)

[Umsetzung 5](#_Toc165378983)

[Ergebnis 5](#_Toc165378984)

[Projektaufbauorganisation 6](#_Toc165378985)

[Zeitplan 7](#_Toc165378986)

[Arbeitsjournal 8](#_Toc165378987)

[Tag 1 – 20.02.2024 8](#_Toc165378988)

[Stundenzettel 8](#_Toc165378989)

[Saldo 8](#_Toc165378990)

[Hilfen 8](#_Toc165378991)

[Fragen 8](#_Toc165378992)

[Reflextion 8](#_Toc165378993)

[Ausblick 8](#_Toc165378994)

[Tag 2 – 27.02.2024 9](#_Toc165378995)

[Stundenzettel 9](#_Toc165378996)

[Saldo 9](#_Toc165378997)

[Hilfen 9](#_Toc165378998)

[Fragen 9](#_Toc165378999)

[Reflexion 9](#_Toc165379000)

[Ausblick 9](#_Toc165379001)

[Tag 3 – 05.03.2024 10](#_Toc165379002)

[Stundenzettel 10](#_Toc165379003)

[Saldo 10](#_Toc165379004)

[Hilfen 10](#_Toc165379005)

[Fragen 10](#_Toc165379006)

[Reflexion 10](#_Toc165379007)

[Ausblick 10](#_Toc165379008)

[Tag 4 – 12.03.2024 11](#_Toc165379009)

[Stundenzettel 11](#_Toc165379010)

[Saldo 11](#_Toc165379011)

[Hilfen 11](#_Toc165379012)

[Fragen 11](#_Toc165379013)

[Reflexion 11](#_Toc165379014)

[Ausblick 11](#_Toc165379015)

[Tag 5 – 14.03.2024 12](#_Toc165379016)

[Stundenzettel 12](#_Toc165379017)

[Saldo 12](#_Toc165379018)

[Hilfen 12](#_Toc165379019)

[Fragen 12](#_Toc165379020)

[Reflexion 12](#_Toc165379021)

[Ausblick 12](#_Toc165379022)

[Tag 6 – 19.03.2024 13](#_Toc165379023)

[Stundenzettel 13](#_Toc165379024)

[Saldo 13](#_Toc165379025)

[Hilfen 13](#_Toc165379026)

[Fragen 13](#_Toc165379027)

[Reflexion 13](#_Toc165379028)

[Ausblick 13](#_Toc165379029)

[Tag 7 – 26.03.2024 14](#_Toc165379030)

[Stundenzettel 14](#_Toc165379031)

[Saldo 14](#_Toc165379032)

[Hilfen 14](#_Toc165379033)

[Fragen 14](#_Toc165379034)

[Reflexion 14](#_Toc165379035)

[Ausblick 14](#_Toc165379036)

[Tag 8 – 02.04.2024 15](#_Toc165379037)

[Stundenzettel 15](#_Toc165379038)

[Saldo 15](#_Toc165379039)

[Hilfen 15](#_Toc165379040)

[Fragen 15](#_Toc165379041)

[Reflexion 15](#_Toc165379042)

[Ausblick 15](#_Toc165379043)

[Tag 9 – 23.04.2024 16](#_Toc165379044)

[Stundenzettel 16](#_Toc165379045)

[Saldo 16](#_Toc165379046)

[Hilfen 16](#_Toc165379047)

[Fragen 16](#_Toc165379048)

[Reflexion 16](#_Toc165379049)

[Ausblick 16](#_Toc165379050)

[Tag 10 – 30.04.2024 17](#_Toc165379051)

[Stundenzettel 17](#_Toc165379052)

[Saldo 17](#_Toc165379053)

[Hilfen 17](#_Toc165379054)

[Fragen 17](#_Toc165379055)

[Reflexion 17](#_Toc165379056)

[Ausblick 17](#_Toc165379057)

[Projektmanagementmethode 18](#_Toc165379058)

[1 Informationsbeschaffung 19](#_Toc165379059)

[1.1 Erwartbarkeit 19](#_Toc165379060)

[1.2 Ziel 19](#_Toc165379061)

[1.3 Unklarheiten 19](#_Toc165379062)

[1.4 Projektumfeld 19](#_Toc165379063)

[1.4.1 Abgrenzung 19](#_Toc165379064)

[1.4.2 Umsysteme 20](#_Toc165379065)

[1.4.3 Assets 20](#_Toc165379066)

[1.4.4 Visual Studio Code 20](#_Toc165379067)

[1.4.5 GitHub 20](#_Toc165379068)

[1.4.5 MongoDB 20](#_Toc165379069)

[1.4.6 Postman 20](#_Toc165379070)

[1.4.7 Konventionen 20](#_Toc165379071)

[1.5 Besprechung 21](#_Toc165379072)

[2 Planung 22](#_Toc165379073)

[2.1 Zeitplan 22](#_Toc165379074)

[2.1.1 Rundung 22](#_Toc165379075)

[2.1.2 Interpretation 22](#_Toc165379076)

[2.1.3 Blocker 22](#_Toc165379077)

[2.1.4 Ist-Zeit 22](#_Toc165379078)

[2.1.5 Meilensteine 22](#_Toc165379079)

[2.2 Testkonzept 23](#_Toc165379080)

[2.2.1 Testmethoden 23](#_Toc165379081)

[2.2.2 Testdaten 24](#_Toc165379082)

[2.2.3 Testprotokoll 24](#_Toc165379083)

[2.3 Testfälle 25](#_Toc165379084)

[2.5 Datensicherung und Versionierung 26](#_Toc165379085)

[2.6 Rest API 26](#_Toc165379086)

[2.6.1 JWT 26](#_Toc165379087)

[2.6.2 Fehlerbehebung 26](#_Toc165379088)

[2.6.3 Mock Daten 26](#_Toc165379089)

[2.7 Daten 27](#_Toc165379090)

[2.7.1 Public Ordner 27](#_Toc165379091)

[MongoDB 27](#_Toc165379092)

[Sicherheit 27](#_Toc165379093)

[2.8 Architektur 27](#_Toc165379094)

[2.9 Node Version 27](#_Toc165379095)

[2.10 27](#_Toc165379096)

[3 Entscheidung 28](#_Toc165379097)

[3.1 Angular Setup 28](#_Toc165379098)

[3.2 Packages 28](#_Toc165379099)

[3.3 Daten 28](#_Toc165379100)

[3.5 Sonstige Konventionen 28](#_Toc165379101)

[4 Realisierung 29](#_Toc165379102)

[4.1 Erstellung Git Repository 29](#_Toc165379103)

[4.2 Angular Setup 29](#_Toc165379104)

[4.2.1 Node Version auswählen. 29](#_Toc165379105)

[4.2.2 Projekt Generieren 29](#_Toc165379106)

[4.2.3 Pakete installieren 29](#_Toc165379107)

[4.2.4 Aufräumen 30](#_Toc165379108)

[4.3 Linting und Testing 30](#_Toc165379109)

[Projektplanung 31](#_Toc165379110)

[User Stories 31](#_Toc165379111)

[Features 31](#_Toc165379112)

[Flowchart 31](#_Toc165379113)

[Mockup 32](#_Toc165379114)

[Glossar 33](#_Toc165379115)

[Abbildungsverzeichnis 34](#_Toc165379116)

[Quellenverzeichnis 35](#_Toc165379117)

# Kurzfassung

### Ausgganssituation

### Umsetzung

### Ergebnis

Teil 1

Dieser erste Teil enthält die Projektaufbauorganisation, den Zeitplan, jegliche Planungsunterlagen und die Arbeitsjournale.

# Projektaufbauorganisation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rolle | Abkürzung | Name | E-Mail | Beschreibung |
| Chefexperte | CEX | Christian Steiner | christian.steiner@kantisargans.ch | Aufsicht über die IPA |
| Hauptexperte | HEX | Christian Steiner | christian.steiner@kantisargans.ch | Hauptansprechpartner für KAND und VF. Begleitet VF und KAND durch die IPA. |
| Nebenexperte | NEX | Christian Steiner | christian.steiner@kantisargans.ch | Stellvertretender Ansprechpartner für KAND und VF während der IPA. Begleitet VF und KAND durch die IPA. |
| Berufsbildner | BB | Christian Steiner | christian.steiner@kantisargans.ch | Verantwortlich für die betriebliche Ausbildung des KAND. |
| Vorgesetzte Fachkraft | VF | Christian Steiner | christian.steiner@kantisargans.ch | Person, welche dem KAND während der IPA als Ansprechperson zur Verfügung steht. Ansprechperson für die Experten dür die jeweilige IPA. |
| Kandidat | KAND | Paul Pietzko | paul.pietzko@stud.kantisargans.ch | Der Lernende, welcher die IPA durchführt. |

# Ein Bild, das Text, Screenshot, Reihe, parallel enthält. Automatisch generierte BeschreibungZeitplan

# Arbeitsjournal

Nachfolgend sind alle Werktage schriftlich in diesem Journal festgehalten. Die im Stundenzettel angegebene Nummer referenziert auf den Zeitplan

## Tag 1 – 20.02.2024

### Stundenzettel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Startzeit** | **Endzeit** | **Zeit** | **Nr.** | **Aufgabe** |
| 13:35 | 14:00 | 25’ | 1.0 | Dokumente anlegen, strukturieren und sichern |
| 14:00 | 15:10 | 70’ | 1.1 | Theorieeinlage |
| Pause |  |  |  |  |
| 15:25 | 15:55 | 30’ | 1.2 | Aufgabenstellung analysieren |
| 15:55 | 16:50 | 55’ | 2.0 | Zeitplan erstellen |
| 16:50 | 17:10 | 20’ | 2.1 | Testfälle schreiben |
| Pause |  |  |  |  |
| 17:20 | 17:30 | 20’ | 7.0 | Arbeitsjournal |

### Saldo

|  |  |  |  |
| --- | --- | --- | --- |
| **Soll** | **Ist** | **Tagessaldo** | **Saldo total** |
| 3.75h | 3.75h | 0 | 0 |

### Hilfen

|  |  |
| --- | --- |
| **Beschreibung** | **Quelle(n)** |
| Informationen zu den Anforderungen und Kriterien. | Herr Steiner (EXP) |
| Layout, Inhalt und Formatierung für Arbeitsjournal, Zeitplan und Testkonzept. | Herr Steiner (EXP) |
| Informationen zu den einzelnen Testmethoden. | chatGPT |

### Fragen

|  |  |  |
| --- | --- | --- |
| **Thema** | **Beschreibung** | **Person** |
| Testen | Ich fragte ob wir testen müssen, wenn wir testing nicht explizit als Kriterium angegeben haben. Die Antwort war «Ja». | Herr Steiner (EXP) |

### Reflextion

Mit meiner heutigen Leistung bin ich sehr zufrieden. Nach der heutigen Theorie von Herr Steiner habe ich ein besseres Verständnis, was alles auf uns zukommen wird. Nach dem heutigen Tag habe ich bereits alle meine Dokumente beisammen, welche ich in Zukunft nur noch ausfüllen brauche. Dies ist eine grosse Hilfe, wenn ich die Dokumente für die richtige IPA in einem Jahr verwenden kann. Denn das Erstellen dieser Dateien hat viel Zeit in Anspruch genommen.

### Ausblick

Nächste Woche Dienstag werde ich das UML erstellen und mit der Realisierung der Applikation beginnen. Wobei ich mich zuerst auf die Backendfunktionalität fokussieren werde.

## Tag 2 – 27.02.2024

### Stundenzettel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Startzeit** | **Endzeit** | **Zeit** | **Nr.** | **Aufgabe** |
| 13:35 | 14:05 | 30’ | 1.1 | Theorieeinlage |
| 14:05 | 15:05 | 60’ | 2.1 | UML Diagramm designen |
| Pause |  |  |  |  |
| 15:20 | 16:30 | 70’ | 2.2 | Mockup erstellen |
| 16:30 | 16:50 | 20’ | 4.0 | Projekt Setup - GitHub, MongoDB |
| Pause |  |  |  |  |
| 17:00 | 17:40 | 40’ | 4.1 | Angular 17, NodeJS |
| 17:40 | 17:50 | 10’ | 7.0 | Arbeitsjournal |

### Saldo

|  |  |  |  |
| --- | --- | --- | --- |
| **Soll** | **Ist** | **Tagessaldo** | **Saldo total** |
| 3.8 | 3.8h | 0 | 0 |

### Hilfen

|  |  |
| --- | --- |
| **Beschreibung** | **Quelle(n)** |
| UML Diagramm realisieren | Herr Steiner (EXP) |

### Fragen

|  |  |  |
| --- | --- | --- |
| **Thema** | **Beschreibung** | **Person** |
| UML – Planung | Die Frage handelte davon, was alles in den User Stories bzw. den Features abgebildet sein muss. | Herr Steiner (EXP) |

### Reflexion

Heute war ebenfalls ein erfolgreicher Tag an dem ich viel erledigen konnte. Meine Zeitplanung und meine ursprüngliche Leitfrage 235 (Entwurf mit UWML) musste ich heute etwas umstrukturieren. Anstelle eines UML Diagrammes habe ich ein Floatchart erstellt und ein Mockup 161 (Entwurf, Design) erstellt. Das hat mich heute mehr Zeit gekostet, als eigentlich vorgesehen. Trotzdem konnte ich bereits mit der Realisation der Applikation starten.

### Ausblick

Für die nächste Woche ist geplant den Home Screen mit allen dazugehörigen Komponenten (wie bsp. der Nav) und das Login/ Signup System anzufangen und fertig zu stellen.

## Tag 3 – 05.03.2024

### Stundenzettel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Startzeit** | **Endzeit** | **Zeit** | **Nr.** | **Aufgabe** |
| 13:35 | 13:55 | 20’ | 1.1 | Theorieeinlage |
| 13:55 | 14:55 | 120’ | 4.1 | Backend |
| 13:55 | 15:10 | 15’ | 7.1 | Expertenbesuch |
| Pause |  |  |  |  |
| 15:25 | 16:30 | 65’ | 6.3 | Dokumentation |
| 16:30 | 16:50 | 20 | 4.1 | Backend |
| Pause |  |  |  |  |
| 17:00 | 17:30 | 30 | 4.1 | Backend |

### Saldo

|  |  |  |  |
| --- | --- | --- | --- |
| **Soll** | **Ist** | **Tagessaldo** | **Saldo total** |
| 3.8 | 2 | 1.8 | 1.8 |

### Hilfen

|  |  |
| --- | --- |
| **Beschreibung** | **Quelle(n)** |
| Wichtigkeit der Dokumentation | Herr Steiner (EXP) |
| Bessere Strukturierung der Arbeit im Ganzen | Herr Steiner (EXP) |

### Fragen

|  |  |  |
| --- | --- | --- |
| **Thema** | **Beschreibung** | **Person** |
| Programmierung | Backendstruktur nach MVC aus vergangen Projekten | Jonas Schmedtmann |

### Reflexion

Bei meinem heutigen Expertenbesuch merkte Herr Steiner an, dass ich mit meiner Planung hinterherhinke und mein IPA Bericht oberste Priorität hat, welche ich bis dato noch nicht erstellt hatte. Darum musste ich den Zeitplan auf die neuen Aktivitäten anpassen.

### Ausblick

In einer Woche werde ich mich wieder vertieft um die Umsetzung und das Programmieren des eigentlichen Projektes kümmern. Zudem werde ich weiter an meinen Unterlagen feilen.

## Tag 4 – 12.03.2024

### Stundenzettel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Startzeit** | **Endzeit** | **Zeit** | **Nr.** | **Aufgabe** |
| 12:50 | 13:05 | 15’ | 1.1 | Theorieeinlage |
| 13:05 | 15:00 | 115’ | 4.1 | Backend |
| Pause |  |  |  |  |
| 15:05 | 15:10 | 5’ | 7.0 | Arbeitsjournal |
| Pause |  |  |  |  |
| 15:25 | 14:00 | 40’ | 4.1 | Backend |
|  |  |  |  |  |

### Saldo

|  |  |  |  |
| --- | --- | --- | --- |
| **Soll** | **Ist** | **Tagessaldo** | **Saldo total** |
| 3.8 | 3 | 0.8 | 2.6 |

### Hilfen

|  |  |
| --- | --- |
| **Beschreibung** | **Quelle(n)** |
|  |  |

### Fragen

|  |  |  |
| --- | --- | --- |
| **Thema** | **Beschreibung** | **Person** |
| Programmierung | Backendstruktur nach MVC aus vergangen Projekten | Jonas Schmedtmann |

### Reflexion

Mit den heutigen drei, vier Stunden habe ich die Backendlogik verbessert. Auch heute kam ich im Zeitplan nicht so weit voran wie geplant, jedoch ist mir nun klarer, wie und was ich die einzelnen Module aus M294 implementieren muss.

### Ausblick

Für die nächsten Lektionen hoffe ich mit dem Backend fertig zu werden. Dies beinhaltet das Login System und das ändern des Kontostandes der User.

## Tag 5 – 14.03.2024

### Stundenzettel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Startzeit** | **Endzeit** | **Zeit** | **Nr.** | **Aufgabe** |
| 7:45 | 08:05 | 20’ | 1.1 | Theorieeinlage |
| 08:05 | 08:35 | 30’ | 4.1 | Backend |
| Pause |  |  |  |  |
| 08:40 | 09:20 | 40’ | 4.1.0 | Backend |
| Pause |  |  |  |  |
| 09:40 | 12:50 | 190’ | 4.1 | Backend |
| Pause |  |  |  |  |
| 13:35 | 15:50 | 115’ | 4.4 | Blackjack game loop programmieren |

### Saldo

|  |  |  |  |
| --- | --- | --- | --- |
| **Soll** | **Ist** | **Tagessaldo** | **Saldo total** |
| 3.8 | 3 | 0.8 | 2.6 |

### Hilfen

|  |  |
| --- | --- |
| **Beschreibung** | **Quelle(n)** |
|  |  |

### Fragen

|  |  |  |
| --- | --- | --- |
| **Thema** | **Beschreibung** | **Person** |
| Programmierung | Backendstruktur nach MVC aus vergangen Projekten | Jonas Schmedtmann |

### Reflexion

Mit den heutigen acht Stunden habe ich einiges erreicht. Die Datenbankverbindung habe ich heute endlich erfolgreich fertig gestellt. Und auch bereits an einigen weitern Features konnte ich bereits weiter- bzw. vorarbeiten. Jedoch ging auch viel zeit für unnötiges debuggen und stumpfes «Copy & Paste» von ChatGPT drauf. So habe ich den Code unnötig verkompliziert und musste mich erstmal wieder zurecht finden. Und den «Spagetti code» entwirren.

### Ausblick

Für das nächste Mal möchte ich das Login System benutzerfreundlicher machen und mit dem Gamloop des Blackjack Spiels anzufangen. Ausserdem möchte ich die Funktionalität des Abziehens und Hinzufügen der Verluste bzw. Gewinne fertig implementieren.

## Tag 6 – 19.03.2024

### Stundenzettel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Startzeit** | **Endzeit** | **Zeit** | **Nr.** | **Aufgabe** |
| 13:35 | 13:40 | 5’ | 1.1 | Theorieeinlage |
| 13:40 | 15:10 | 90’ | 4.5 | Slots Machines programmieren |
| Pause |  |  |  |  |
| 15:25 | 17:50 | 115’ | 4.40 | Blackjack game loop programmieren |

### Saldo

|  |  |  |  |
| --- | --- | --- | --- |
| **Soll** | **Ist** | **Tagessaldo** | **Saldo total** |
| 3.8 | 3.8 | -2.6 | 0 |

### Hilfen

|  |  |
| --- | --- |
| **Beschreibung** | **Quelle(n)** |
|  |  |

### Fragen

|  |  |  |
| --- | --- | --- |
| **Thema** | **Beschreibung** | **Person** |
| Programmierung | Lösungsansätze zur Programmierung der Funktionalitäten. | OpenAI, Chat GPT |
| Programmierung | Angular Konventionen | OpenAI, Chat GPT |

### Reflexion

Diesen Nachmittag konnte ich den Funktionalität des Praktischen Teil dieser Arbeit beinahe fertigstellen. Nach Donnerstag letzer Woche, konnte ich all die angefangen Funktionen miteinander verbinden und zum laufen bringen. Diese Woche bin ich auch viel ruhiger an die Sache herangegen. Was mir dabei half mich mehr in den Programmablauf hineinzuversetzen und somit besser zu debuggen.

### Ausblick

Für den nächsten Arbeitstag nehme ich mir vor zum einen Die Dokumentation weiterzuschreiben und den geschrieben Code zu finalisieren. Anschliessend werde ich den code «Refraktoren» und überarbeiten.

## Tag 7 – 26.03.2024

### Stundenzettel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Startzeit** | **Endzeit** | **Zeit** | **Nr.** | **Aufgabe** |
| 13:35 | 15:10 | 95’ | 6.3 | Dokumentation |
| Pause |  |  |  |  |
| 15:25 | 17:50 | 115’ | 6.3 | Dokumentation |

### Saldo

|  |  |  |  |
| --- | --- | --- | --- |
| **Soll** | **Ist** | **Tagessaldo** | **Saldo total** |
| 3.8 | 3.8 | 0 | 0 |

### Hilfen

|  |  |
| --- | --- |
| **Beschreibung** | **Quelle(n)** |
| Wichtigkeit der Dokumentation | Herr Steiner (EXP) |
| Allgemeine Verbesserungen in Dokumentationsprozess | Herr Steiner (EXP) |

### Fragen

|  |  |  |
| --- | --- | --- |
| **Thema** | **Beschreibung** | **Person** |
|  |  |  |

### Reflexion

Den Zeitplan habe ich heute nicht eingehalten, da ich den Fokus der heutigen Lektionen mehr auf die Dokumentation legen wollte. Die letzten Nachmittage habe ich alle für das Programmieren meiner Arbeit genutzt. Die nun alle Grundfunktionen korrekt beinhaltet. Für heute wäre eigentlich refactoring angesagt gewesen. Jedoch war ich mit meiner Dokumentation vor heute fast noch nirgends. Zudem hat das zweite Expertengespräch nochmals deutlich gemacht, wie wichtig die Dokumentation zum jetzigen Standpunkt bereits ist. Und darum gingen alle fünf Lektionen dafür drauf. Ich habe fast alles aus der Doku gelöscht und nochmals erfolgreich frisch angefangen.

### Ausblick

Beim nächsten Mal werde ich mich wieder an die Dokumentation setzten und versuchen sie weiter zu vervollständigen. Und nach dem ich das Grundgerüst der Dokumentation habe, werde ich mich an das Verfeinern des Codes wagen.

## Tag 8 – 02.04.2024

### Stundenzettel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Startzeit** | **Endzeit** | **Zeit** | **Nr.** | **Aufgabe** |
| 13:35 | 15:10 | 95’ | 6.3 | Dokumentation |
| Pause |  |  |  |  |
| 15:25 | 16:00 | 35’ | 6.3 | Dokumentation |

### Saldo

|  |  |  |  |
| --- | --- | --- | --- |
| **Soll** | **Ist** | **Tagessaldo** | **Saldo total** |
| 3.8 | 3.8 | 0 | 0 |

### Hilfen

|  |  |
| --- | --- |
| **Beschreibung** | **Quelle(n)** |
|  |  |

### Fragen

|  |  |  |
| --- | --- | --- |
| **Thema** | **Beschreibung** | **Person** |
|  |  |  |

### Reflexion

Heute habe ich, wie letzte Woche angekündigt, an der Dokumentation weitergearbeitet. Jedoch ging es mir nach dem Mittag nicht gut, weshalb ich schon um 16:00 Uhr anstatt 17:50 Uhr nach Hause gegangen bin. Diese 110’ Minuten werde ich darum zuhause nachholen müssen. Dies ist jedoch nicht allzu schlimm, da für heute nur die Dokumentation geplant war.

### Ausblick

Beim nächsten Mal werde ich das Schreiben der Dokumentation pausieren und mich wieder an das Programmieren wagen. Denn die Applikation muss noch nach dem Testprotokoll getestet werden.

## Tag 9 – 23.04.2024

### Stundenzettel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Startzeit** | **Endzeit** | **Zeit** | **Nr.** | **Aufgabe** |
| 13:35 | 15:10 | 95’ | 6.3 | Dokumentation |
| Pause |  |  |  |  |
| 15:25 | 17:50 | 145’ | 6.3 | Dokumentation |

### Saldo

|  |  |  |  |
| --- | --- | --- | --- |
| **Soll** | **Ist** | **Tagessaldo** | **Saldo total** |
| 3.8 | 3.8 | 0 | 0 |

### Hilfen

|  |  |
| --- | --- |
| **Beschreibung** | **Quelle(n)** |
| Fachwörter lernen | Herr Steiner (EXP) |

### Fragen

|  |  |  |
| --- | --- | --- |
| **Thema** | **Beschreibung** | **Person** |
|  |  |  |

### Reflexion

Die fünf Lektionen habe ich heute wieder komplett in die Erstellung und Nachtragung der Dokumentation gesteckt. Während des heutigen Expertengespräch wurde mir nochmals gesagt, wie wichtig es ist die Dokumentation zeitgleich der Programmierung zu erstellen. Dies habe ich nicht gemacht, da mir die Dokumentation und die Anforderungen an sie zu dem damaligen Zeitpunkt noch zu viel neues auf einmal waren.

### Ausblick

Es sin schon 80% der Arbeit vorbei. In der Schule werde ich den Fokus nur noch auf die Dokumentation legen und in Eigenarbeit zuhause die fehlenden Testfälle schreiben und den Code finalisieren. Wobei das nicht im Vordergrund steht.

## Tag 10 – 30.04.2024

### Stundenzettel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Startzeit** | **Endzeit** | **Zeit** | **Nr.** | **Aufgabe** |
| 13:35 | 13:55 | 20’ | 1.1 | Theorieeinlage |
| 13:35 | 15:10 | 75’ | 6.3 | Dokumentation |
| Pause |  |  |  |  |
| 15:25 | 17:50 | 145’ | 6.3 | Dokumentation |

### Saldo

|  |  |  |  |
| --- | --- | --- | --- |
| **Soll** | **Ist** | **Tagessaldo** | **Saldo total** |
| 3.8 | 3.8 | 0 | 0 |

### Hilfen

|  |  |
| --- | --- |
| **Beschreibung** | **Quelle(n)** |
| Fachwörter lernen | Herr Steiner (EXP) |

### Fragen

|  |  |  |
| --- | --- | --- |
| **Thema** | **Beschreibung** | **Person** |
|  |  |  |

### Reflexion

### Ausblick

Teil 2

In diesem zweiten Teil der Dokumentation wird das Projekt von der Aufgabenstellung bis zur Auswertung beschrieben.

# Projektmanagementmethode

Eine passende Projektmanagementmethode ist die Voraussetzung einer effizienten Entwicklung. Die gängigsten Varianten und die von mir benutze werden unten aufgelistet.

Vorgehensmodelle können in iterative und sequenzielle Arten unterteilt werden. Iterative Modelle, wie beispielsweise «Scrum», werden in der Softwareentwicklung meist bevorzugt. Sie ermöglichen die schrittweise Entwicklung und Auslieferung von kleinen Teilen der Software in kurzen Zyklen. Hat man aber zu Beginn bereits ein vollständiges Ziel, arbeitet allein oder will den Fokus auf die Dokumentation legen, ist dies wohl nicht der ideale Ansatz. In solchen Fällen wählt man besser eines der sequenziellen Vorgehensmodelle, bei welchen Wert auf klare aufeinanderfolgende Phasen gelegt wird. Dabei wird mit der nächsten Tätigkeit erst begonnen, wenn der vorherige Schritt vollständig abgeschlossen ist. Im Fokus steht eine intensive Planungsphase, welche Unklarheiten und Veränderungen während der Umsetzung minimieren soll. Auch eine strenge Kontrolle und hochwertige Dokumentation wird bei den meisten dieser Modelle ermöglicht.

Die Entscheidung fiel auf das sequenzielle IPERKA Modell. Wenn sauber nach IPERKA gearbeitet wird, können alle Punkte aus der Vorgabe abgedeckt werden.

Bei diesem Modell wird ein Projekt in sechs Phasen unterteilt:

* **Informieren**: In einem ersten Schritt analysiert man den Auftrag, sodass man diesen genauestens versteht.
* **Planen**: Daraufhin beginnt man sich Gedanken über mögliche Lösungsvarianten zu machen.
* **Entscheiden**: Danach fällt man Entscheidungen zu den einzelnen Varianten.
* **Realisieren**: Im vierten Schritt beginnt man mit der Umsetzung des Projektes.
* **Kontrollieren**: Später wird überprüft, ob der Auftrag korrekt umgesetzt wurde.
* **Auswerten**: Als Letztes wird reflektiert, wie die Arbeit verlief.

Ein Nachteil hat IPERKA. Fehler werden erst nach der kompletten Umsetzung erkannt. So ist es schwierig, auf diese zu reagieren. Deshalb ist es ratsam schon während der Realisierung die einzelnen Komponenten zu testen, bevor sie als erledigt angesehen werden. In der Kontrollphase wird dann die Applikation im Komplettpaket auf alle Kriterien überprüft.

Neben IPERKA gibt es auch andere sequenzielle Projektmanagementmethoden. Das Wasserfall- oder V-Modell wären zwei grosse Vertreter solcher Managmentmethoden. Da jedoch IPERKA häufiger in schulischen Zwecken verwendet wird. Findet sie nun auch hier besseren Anklang.

# 1 Informationsbeschaffung

Als Erstes musste verstanden werden, warum es nötig ist den Auftrag zu erfüllen. Dies ermöglicht es, die Sicht des Auftraggebers besser nachzuvollziehen.

Online Casinos sehen von Aussen betrachtet sehr komplex und eindrucksvoll aus, sind jedoch in ihrem Kern einfach umzusetzen. Zudem ist es ein perfektes Projekt für eine Multiuserapplikation mit festgelegten Zielen und nicht Zielen. Die Applikation wurde mit dem JavaScript Framework Angular umgesetzt, da dies ein sehr robustes Framework ist. Die App beinhaltet keine komplexe Animationen, weshalb sie gut mit Angular umzusetzen ist.

## 1.1 Erwartbarkeit

Die Arbeit soll starken Wert auf die Umsetzung mit Clean Code, Kiss und weiteren coding Konventionen wie MVP setzten. Codedateien sollen in eine verständlich gute Struktur unterteilt werden.

## 1.2 Ziel

All die gelernten coding Konventionen aus den drei Jahren Informatik Unterricht an der Kantonsschule Sargans sollen sauber umgesetzt werden, so dass der Code auch in Zukunft gut wartbar und erweiterbar ist.

## 1.3 Unklarheiten

Wie soll die Backendstruktur aussehen. Zuerst war eine verknüpfte Backend mit Frontend Variante angedacht. Wobei Die SSR Funktionalität von Angular so genutzt wird, dass alles auf einem Server läuft.

Jedoch wurde durch die Leitfarge 249 – MVC (Programmierung) diese Überlegung über den Haufen geworden.

## 1.4 Projektumfeld

Das Projektumfeld beinhaltet die Gesamtheit aller internen und externen Faktoren, die das Projekt beeinflussen können. Es umfasst die organisatorischen Rahmenbedingungen, die technische Infrastruktur, die beteiligten Personen und Gruppen, deren Interessen und Einflüsse, sowie die relevanten Markt- und Umweltbedingungen.

### 1.4.1 Abgrenzung

Die folgend aufgelisteten Themen gehören nicht zum Umfeld dieser IPA Arbeit.

#### Design

Das Design wurde nicht von Grund auf neu erfunden, sondern wurde von bekannten Casino Seiten wie «Jackpots.ch» abgeleitet.

#### Assets

Einige Illustrationen wie beispielsweise die Spielkarten SVGs wurden von GitHub repositories entnommen, da das erstellen solcher Dateien unnötig Zeit in Anspruch nehmen würde.

#### Testing Systeme

Die eingesetzten Test-Frameworks Jasmine und Karma wurden eingesetzt schneller und qualitativ besser testen zu können.

### 1.4.2 Umsysteme

Einige Funktionen wurden als library als npm install in das Projekt heruntergeladen. Bycript und JWT sind Beispiele, die in Eigenentwicklung ebenfalls zu viel Zeit in Anspruch genommen hätten den Ramen dieser Arbeit masslos gesprengt hätten.

### 1.4.3 Assets

Für dieses Projekt wurde eine SVG Sammlung von Pokerkarten importiert. Zudem wurde in Eigen Kreation eine Sammlung von Jeton

### 1.4.4 Visual Studio Code

Visual Studio Code wurde als DIE verwendet, da sie Industrie Standard ist.

Dazu wurden einige Add-Ons in VS Code installiert um den Entwicklungsprozess zu vereinfachen. Dazu gehören grössten Teils Add-Ons von den Entwicklern der Technologien, wie MongoDB, Angular, NodeJS usw.

### 1.4.5 GitHub

Zur Versionierung, Synchronisierung und Aufbewahrung des Codes dient GitHub für dieses Projekt. GitHub wird ebenfalls an der KSS als Versionsverwaltungtool standardmässig eingesetzt.

Während der ganzen Entwicklung wird nur auf dem Main Branch gearbeitet, da das Projekt von nur einer Person entwickelt wird und nicht veröffentlicht wird.

### 1.4.5 MongoDB

Für das Speichern von Nutzerdaten dient MongoDB; eine weit verbreitete Datenbanklösung welche das Speichern in JSON Format ermöglicht, was wiederrum mit der Programmierung von JavaScript harmoniert.

### 1.4.6 Postman

Als Testmittel der zu erstellenden REST-API wird Postman eingesetzt. Dies ermöglicht es per JSON Anfragen an einen definierten API-End Point Anfragen zu senden und Responses auf dessen Korrektheit zu prüfen.

### 1.4.7 Konventionen

Im Auftrag wurde beschrieben, dass die allgemeinen wie auch zusätzliche Leitfragen Konventionen eingehalten werden sollen, um den im Projektauftrag beschrieben Clean Code Standard zu halten.

#### Ein Bild, das Text, Screenshot, Karte Menü, Schrift enthält. Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Schrift, Design enthält. Automatisch generierte BeschreibungStruktur

Abbildung 2: Backend Strucktur

Abbildung 1: Backend Strucktur

#### Commits

Commits werden immer nach dem Datum des geamchten Tages benannt. Dabei wird immer am Ende des Arbeitstages committed, es sei denn es gibt viel Code der auf einmal geändert wird. Dann werden Mehrere Commits an einem Tag gemacht, welche mit v +Nummer (v1, v2..) benannt werden.

**Beispiel**:  
*26.03.2024 v1*

#### Branches

Der ganze Entwicklungsprozess findet alleine auf dem Default Main Branch statt.

## 1.5 Besprechung

Um sicherzustellen, dass alle Anforderungen an das Projekt korrekt formuliert waren, wurde das Projekt abwechselnd mit dem HEX besprochen und angepasst.

# 2 Planung

## 2.1 Zeitplan

Nach der Informationsbeschaffung und einer genauen Analyse konnte der Zeitplan erstellt werden. Das Gantt Diagramm ist Teil 1 unter «Zeitplan» ersichtlich.

### 2.1.1 Rundung

Vor der Erstellung wurde definiert, wie detailliert die Aufgaben geplant werden sollen. Die Aufteilung besteht jeweils aus Paketen, die jeweils nicht länger als vier Stunden sein dürfen. Da es aber auch viele kleine Aufgaben gibt, wurden diese manche mit anderen Aufgaben zusammengefügt und dessen Zeitaufwand nach bestem Wissen geschätzt.

### 2.1.2 Interpretation

Für die Aufgaben während der Realisierung wurde Zeit für die Dokumentation inkludiert. Wenn also für eine Funktion 3 Stunden geplant wurden, darf ein Teil dieser Zeit auch zur Dokumentierung des Vorgehens genutzt werden.

### 2.1.3 Blocker

Für die Informationsbeschaffung und Erstellung des Zeitplans wurde keine genaue Zeit geschätzt. Dies liegt an IPERKA. Schliesslich konnte erst bei der Planungsphase geplant werden. Zu beginn wurden aber extra Zeit blockiert.

### 2.1.4 Ist-Zeit

An jedem Arbeitstag werde ich am Schluss die Zeiten eintragen und einen Soll-Ist-Vergleich vornehmen. Dieser wird dann im Arbeitsjournal festgehalten. Die im Arbeitsjournal eingetragenen Zeiten werden immer auf fünf Minuten gerundet und in Minuten angegeben.

### 2.1.5 Meilensteine

Diese Meilensteine wurden zu Beginn der Arbeit definiert. Um den Soll-Ist-Vergleich besser abgleichen zu können.

|  |  |  |
| --- | --- | --- |
| **Meilenstein** | **Tag der geplanten Erreichung** | **Tag der Erreichung** |
| Angular Anwendung Mit allen Screens | 27.02 | 27.02 |
| Rest API mit allen Endpoints | 05.03 | 14.03 |
| Blackjack Game Loop | 12.03 | 14.03 |
| Slots Maschinen Game Loop | 14.03 | 26.03 |
| Refactoring | 26.03 | 16.04 |
| Dokumentation | 16.04 |  |

## 2.2 Testkonzept

Getestet wird, um sicherzustellen, dass eine Software-Anwendung korrekt funktioniert, Benutzeranforderungen erfüllt und frei von Fehlern ist, die die Nutzung beeinträchtigen könnten, sowie um das Risiko von Problemen nach der Veröffentlichung zu minimieren und dadurch die Qualität und Zuverlässigkeit des Produktes zu gewährleisten.

### 2.2.1 Testmethoden

* **Blackbox-Test**: Hierbei kennt der Tester den internen Aufbau oder den Code der Anwendung nicht und testet nur die Benutzeroberfläche und die Funktionalitäten, um sicherzustellen, dass die Anforderungen erfüllt sind.
* **Whitebox-Test**: Im Gegensatz zum Blackbox-Test hat der Tester Zugang zum internen Code und kann so die internen Strukturen testen, um die Abdeckung des Codes sicherzustellen, beispielsweise durch Pfad- oder Zweigtests.
* **Unit-Test**: Dabei werden einzelne Komponenten oder Funktionen der Software isoliert getestet, um ihre korrekte Funktionsweise zu verifizieren.
* **Integrationstest**: Hierbei werden mehrere Komponenten oder Systeme kombiniert und als Gruppe getestet, um zu überprüfen, ob sie korrekt zusammenarbeiten.
* **Systemtest**: Bei dieser Art des Tests wird das gesamte, vollständig integrierte System auf Übereinstimmung mit den spezifizierten Anforderungen geprüft.
* **Akzeptanztest**: Dieser Test wird durchgeführt, um festzustellen, ob das System die Geschäftsanforderungen erfüllt und für den Einsatz beim Endbenutzer bereit ist.

#### Jasmine

Mit Jasmine können Behavior-Driven Development (BDD) Tests für JavaScript-Anwendungen erstellt werden. Es legt Wert darauf, dass das Verhalten einer Funktion genau wie spezifiziert überprüft wird. Einmal implementiert, helfen die Tests Entwicklern zu verstehen, ob das Verhalten der Funktion auch nach Änderungen noch dem vorgegebenen Muster entspricht. Weitere Informationen zu Jasmine sind auf der Webseite: https://jasmine.github.io/ verfügbar.

#### Karma

Karma ist ein Test-Runner, der für das Ausführen von JavaScript-Tests direkt im Browser entwickelt wurde. Dies ermöglicht es, dass Tests unter realen Bedingungen ausgeführt werden können. Nachdem die Testfälle implementiert sind, wird überprüft, ob die Ergebnisse den Erwartungen entsprechen, was gerade bei Cross-Browser-Testing von Vorteil ist. Mehr zu Karma findet man unter: https://karma-runner.github.io/.

#### Whitebox

Beim White-Box-Test werden interne Strukturen einer Anwendung getestet. Hierbei geht es darum, die internen Wege, die Logik und die Implementierungsdetails des Codes zu überprüfen. Ein Test wird so implementiert, dass er die interne Funktionsweise überprüfen und sicherstellen kann, dass alle Pfade und Zweige des Codes unter verschiedenen Bedingungen funktionieren. Dies erfordert tiefes Verständnis des zu testenden Systems.

#### Schlussfolgerung

Wie aus den Test-Methoden bzw Test Frameworks ersichtlich wird. Wird das Projekt auf zwei Arten getestet. Zum einen Automatische Test, die direkt im Browser ausgeführt werden, als auch per Hand aus Entwicklersicht.

* **Automatisch**

Manche Test werden vom Programm während der Laufzeit automatisch ausgeführt. Diese werden im Vorfeld definiert und führen sich dann von alleine immer wieder selbst aus.

* **Manuell**

Zudem werde ich einzelnen Funktionen per Hand im ausgeführten Programm destruktiv testen.

Bei jedem Testfall ist in der Spalte «Art» jeweils einer dieser zwei Begriffe angegeben. Dadurch ist nachvollziehbar, in welcher Umgebung die Tests durchgeführt wurden.

Hinweis: Details zum Vorgehen werden nicht getestet. So werden beispielsweise die Einhaltung der Commit Conventions nicht getestet, da diese später unter «Auswertung» überprüft werden.

### 2.2.2 Testdaten

Es werden die erhobenen Daten der Benutzer ausgewertet. Ansonsten muss nichts beachtet werden.

### 2.2.3 Testprotokoll

Getestet wird auf Windows 11 mit Opera als Browser. Dabei werden die Testing-Frameworks Jasmine in der Version 3.9.0 und Karma in der Version 6.3.2 Verwendet. Destruktive Whitebox Tests werden sowohl im Frontend als auch im Code durchgeführt. Dabei wird die Datenbank des Projektes MongoDB auf ihre Korrektheit geprüft. Der zu testende Code im Backend ist Node.js in der Version 14.17.0 und im Frontend Angular in der Version 17.0.0. Der Tester wird dabei der Kandidat KAND sein.

## 2.3 Testfälle

Nachfolgenden sind alle Testfälle definiert. Diese decken alle relevanten Kriterien des Auftrages ab.

#### Basisarbeiten

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr.** | **Art** | **Was wird getestet?** | **Erwartetes Ergebnis** |
| 1.0 | Whitebox-Tests  (Jasmine) | Registrierung eines neuen Benutzers | Eingabe von validen Benutzerdaten |
| 2.0 | Whitebox-Tests  (Jasmine) | Login-Funktionalität | Eingabe von gültigem Nutzernamen und Passwort |
| 3.0 | Whitebox-Tests  (Jasmine) | Aktualisierung des Kontostandes | Ergebnis eines Slotmaschinenspiels (Gewinn oder Verlust) |
| 4.0 | Unit-Tests (Karma) | Ergebnisse eines Slotmaschinen-Spins | Durchführung eines Spins mit Einsatz |
| 5.0 | Unit-Tests (Karma) | Spiellogik von Blackjack | Durchführung einer Hand Blackjack |
| 6.0 | Unit-Tests (Karma) | Validierung von Benutzerdaten bei Registrierung | Eingabe von fehlerhaften Benutzerdaten |

## 2.5 Datensicherung und Versionierung

Es ist wichtig, dass Daten vor Verlust Geschützt sind. Zudem ist auch eine Versionierung von Nöten, um auf älteren Code zugreifen zu können Der Sourcecode wird dementsprechend auf GitHub abgespeichert und ist so versioniert und protokolliert. Lokale Änderungen werden dazu zu sinnvollen Zeitpunkten gepusht. Die Dokumentation und der Zeitplan werden ebenfalls auf Vorgabe des HEX in das GitHub Repository hochgeladen.

## 2.6 Rest API

Um die Kommunikation zwischen der Anwendung und der Datenbank zu gewährleisten wurde mithilfe von Node.js eine RestAPI erstellt. Die API beinhaltet die folgenden Endpunkte und funktionen. Siehe Abbildung 2.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 2: API Endpoints

### 2.6.1 JWT

JSON Web Token ist ein offener Standard, der eine kompakte und selbständige Methode für sicher übermittelte Informationen zwischen Parteien als JSON-Objekt bietet. In dieser Anwendung werden JWTs verwendet, um die Authentifizierung von Benutzern zu verwalten.

### 2.6.2 Fehlerbehebung

Eine API ist eine weitere Abhängigkeit die korrekt konfiguriert sein, da sonst die Applikation im ganzen nicht Funktioniert. Die Applikation würde unter Umständen bei Ausfällen der API nicht funktionieren. Dies ist unter folgenden Umständen der Fall:

* Bei einem Ausfall des hostenden Servers
* Falls die Internetverbindung des Rechners ausfällt

In solchen Fällen müssen den Benutzern entsprechende Fehlermeldungen angezeigt werden.

### 2.6.3 Mock Daten

Um während der Entwicklung den geschriebenen Code testen zu können wurden einige fiktive Benutzer angelegt. Diesen wurde über die Zeit immer wieder Geld übertragen, um die Gewinne und Verluste überprüfen zu können Mockdaten sind vorgefertigte Werte, welche genutzt werden können, um ein Verhalten zu simulieren.

## 2.7 Daten

Da die Applikation Nutzerdaten sammelt um funktionieren zu können, stellt sich die Frage, wo diese gelagert werden und wie man ihre Sicherheit garantieren kann.

### 2.7.1 Public Ordner

In Angular gibt es den «public» Ordner. Dieser ist angedacht um leifht pflegbare Daten, wie Illustrationen und andere Frontendwichtige Dateien zu lagern. Diese Daten werden dann mit der Applikation als ganzes auf einen fremden Server Deployed, wenn die Absicht besteht, die Applikation zu veröffentlichen.

### MongoDB

Die gesamten gesammelten Nutzerdaten werden über das erstellte Backend an das verbundene MongoDB Cluster gesendet. Dort werden sie in der Collection «Users» in der «data» Datenbank gespeichert.

### Sicherheit

Die Passwörter werden nur gehashed gespeichert. Dies verhindert unerlaubten Zugriff auf Nutzerkonten nach einer Attacke auf die Applikation.

## Model–view–controller - Wikipedia2.8 Architektur

Abbildung 3: MVC Architektur

Dieses Projekt hält sich an die MVC (Model, View, Controller) Architektur. Dazu wird die Software in drei Schichten «Model», «View» und «Architektur» unterteilt. Jede Schicht ist für eine Aufgabe zuständig und kommuniziert mit der direkt darüber- oder darunterliegenden Schicht. Dadurch werden die Verantwortlichkeiten klar getrennt, weshalb ganze Schichten auch leicht ausgetauscht werden können.

Das Backend wird dabei in Models und Controllers unterteilt. Jedoch dient das Backend nur als API also als Schnittstelle von Datenbank zum Frontend. So wird im Angular Code nicht nur das Styling sondern auch die Logik der Applikation gehandhabt. In Angular Selber wird dabei auf die Konventionen zurückgegriffen die beim Umgang mit dem Framework üblich sind.

## 2.9 Node Version

Wichtig bei der Installation ist, dass eine kompatible Node-Version verwendet wird. Die Anforderungen in diesem Projekt sind Versionen ab 18.13.0 und neuer unterstützt.

## 2.10

In diesem Projekt werden folgende Schriften verwendet:

* Roberto
* Arial

# 3 Entscheidung

In der dritten Phase von IPERKA werden Entscheidungen getroffen.

## 3.1 Angular Setup

Wie ein Angular Projekt generiert werden kann, steht in folgender Anleitung beschrieben:  
*https://angular.io/tutorial/tour-of-heroes/toh-pt0*

Bei der Initialisierung dieses Projektes wurden folgende Schritte getätigt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Abfrage** | **Erklärung** | **Möglichkeiten** | **Entscheid** | **Begründung** |
| Prefered stylesheet format | Verschiedene Arten von CSS stehen zur Auswahl | CSS, SCSS, Sass, Less | SCSS | Leicht verständlich und Schlanker als basis CSS |
| ServerSide Rendering | Soll die Applikation Server Side gerendert werden | Yes/ No | Yes | ServerSide Rendering hat viele vorteile gegenü. ClientSide Rendering |

## 3.2 Packages

Als Programmierer muss man nicht alles von Grund auf selbst entwickeln. Es gibt Codegerüste, welche ein anderer Programmierer bereits vollends ausprogrammiert hat und anderen in der Branche gratis zur Verfügung stellt.

Dieses Projekt beinhaltet Packages von mehreren externen Technologien. Vermehrt sind Angular spezifische packages installiert, die zum Teil vorausgesetzt sind und zum anderen viele Aufgaben vereinfachen.

Zudem sind Backend spezifische packgages wie cors, dotenv, morgen und express wie auch mongoose für die Datenbank vorhanden.

Um den Code besser abstrahieren zu können, wird RxJS verwendet. Diese library ermöglicht es Daten in einer deklarativen Art zu manipulieren.

## 3.3 Daten

In der Planung wurden Methoden zu Speicherung von Daten aufgeführt. In diesem Abschnitt wird nun der Entscheid für die Implementierung der Route dargelegt. Unter «api/v1/users» werden die Nutzerdaten für das Frontend, mit erfolgreicher Authentifizierung, als JSON zur Verfügung stehen.

Es stellt sich jedoch noch die Frage, ob alle Daten auf einmal gefetchet und in den Komponenten manipuliert oder der Service die Logik bei sich hält und nur bestimmte Daten zurückliefert. Die Lösung sind unterschiedliche Endpoints von der API. Die entsprechenden Komponenten, fetchen also immer nur die Daten, die sie auch wirklich brauchen. So kann ein Overlord von Daten unterbunten und die Leistung gesteigert werden.

## 3.5 Sonstige Konventionen

Im Projekt werden alle von Angular geforderten Konventionen korrekt umgesetzt. So werden einzelne Komponenten je nach grösse weiter unterteilt und einzelne Funktionen in Services outgesourcet .

# 4 Realisierung

Nach dem die Entscheidungen getroffen wurde, konnte die Planung realisiert werden.

## 4.1 Erstellung Git Repository

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte BeschreibungMit der Initialisierung einer Angular Anwendung wird automatisch ein lokales Git Repo erstellt. Dieses kann dann entweder durch GitHub Desktop oder direkt über Visual Studio Code direkt auf GitHub veröffentlicht werden. Die Sichtbarkeit dieses Projektes ist Privat, wobei der HEX als Kollaborator auf das Projekt Einsicht hat. Zudem werden die Dokumente wie der Zeitplan und der IPA-Bericht jede Woche, entsprechend den commit Konventionen, mit dem Code gepusht.

Abbildung 4: Erstellung Repository

Nachdem das Projekt erstellt wurde, wurde wie beschrieben Der HEX als Kollaborator dem Projekt hinzugefügt.

Danach konnte das Repository im Terminal mit dem folgenden Befehl geklont werden:

*git clone https://github.com/paulpietzko/casino-app.git*

## 4.2 Angular Setup

Das Setup wurde wie geplant durchgeführt. Dazu waren vier Schritte notwendig.

### 4.2.1 Node Version auswählen.

Bei der Initialisierung wurde automatisch die neuste (@Latest) kompatible Version von Node ausgewählt.

### 4.2.2 Projekt Generieren

Das Projekt wurde dann entsprechend dem beschrieben Setup Prozess (siehe 3.1 Angular Setup) generiert.

### 4.2.3 Pakete installieren

Wie im Kapitel «Entscheidungen» beschrieben wird, werden noch weitere NPM-Pakete im Projekt benötigt. Diese zwei Befehle installierten beispielsweise mongoose und bycript

npm i mongoose  
npm i bycript

### 4.2.4 Aufräumen

Zum Schluss musste das Projekt noch etwas aufgeräumt werden. Schliesslich sollen keine unrelevanten Dateien und Konfigurationen das Repository überfüllen.

Um den Projektumfang nicht zu gross zu machen, wurde im Projekt nur unbedingt notwendige libraries und Operationen verwendet bzw. installiert.

## 4.3 Linting und Testing

Das Ziel ist, einen gleichmässigen Code zu schreiben. Dafür müssen Regeln definiert werden, nach denen sich die Entwickler richten müssen. Dies ist die Voraussetzung für einen sauberen Code. Beim Linting während des Deploy-prozesses überprüft, ob diese Regeln eingehalten wurden.

Jedoch wurde in diesem Projekt kein spezifischer Linter verwendet, da aus Zeit- und Managmentgründen der Frontend Code mit dem Backendcode zusammengelegt wurde. Durch den EXP wurde festgelegt, dass das Projekt nur auf einem Repository angelegt werden soll. Durch diese Struktur liessen sich keine Linter konfigurieren, da diese entweder für den Backendcode (Node) oder für das Frontend (Angular) konfiguriert werden müssen.

Folgende Linter und CI/CD Pipeline Tools währen mit einer anderen Projektstruktur kompatibel.

|  |  |
| --- | --- |
| **Tool** | **Beschreibung** |
| ESLint | Analysiert JavaScript, TypeScript; erzwingt Code-Standards |
| Staged Linter | Prüft geänderte Dateien vor Commit; verwendet Husky, lint-staged |

### 4.3.1 Prettier

Ein Bild, das Text, Screenshot, Karte Menü, Schrift enthält.

Automatisch generierte BeschreibungPrettier ist dazu da, den Code nach Vorgaben zu formatieren. Mit dem Befehl npm run prettier wird der Gesamte code geprüft und angepasst. Mehr über das Tool erfährt man auf der offiziellen Website: *https://prettier.io/* . Dieses Tool nimmt direkte Änderungen am Code vor. Dabei wird jede Zeile nach den Vorgaben formatiert. Beispielsweise wird jeder Zeile ein Semikolon angefügt, wenn man dies vorgibt

Abbildung 5: Prettier

## 4.5 CI/CD

Continuous Integration/ Continuous Deployment ist ein Prozess, bei welchem der Code regelmässig und automatisch getestet, gebuildet und auf ein Repository oder sogar eine Produktionsumgebung geladen wird. Durch die Automatisierung wird sichergestellt, dass der Code jederzeit fehlerfrei ist.

Aus Zeitgründen wurde auf eine CI Pipeline verzichtet. Und da das Projekt nicht real deployed wird, konnte auch auf eine CD Pipeline verzichtet werden.

## 4.6 Branches

Da nicht geplant war auf einem anderen Branch als dem Main Branch zu arbeiten, konnte das Git direkt benutzt werden. Dies war möglich, da nur der KAND alleine an dem Projekt gearbeitet hat und keine CD/CD Integration vorhanden war.

## 4.7 Backend

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungWährend des Informatikunterricht an der IMS wurden alle Web Applikationen mit ein und der gleichen Struktur nach MVC programmiert. Deshalb wurde diese Struktur in diesem Projekt übernommen.

### 4.7.1 Server.JS

Dabei wird in der server.jseine einfache Verbindung zum MongoDB Cluster erstellt.

### 4.7.2 App.JS

In der app.js wird die Middleware definiert. Und beispielsweise Cors für alle Endpoints aktiviert.

### 4.7.3 AppError

Im «utils» folder wird eine benutzerdefinierte Fehlerklasse AppError definiert, die von der eingebauten  
JavaScript-Klasse Error erbt.

### 4.7.4 Routes

Unter «routes» werden die Routen für Benutzeroperationen definiert. Es wird ein Router-Objekt von Express verwendet, um spezifische HTTP-Methoden (GET, POST, PATCH, DELETE) mit entsprechenden Controller-Funktionen zu verknüpfen.

Abbildung 6: Backend Struktur

### Ein Bild, das Text, Screenshot enthält. Automatisch generierte Beschreibung4.7.5 User Schema

Abbildung 7: User Schema

Damit alle Benutzer einheitlich angelegt werden können, wird ein Mongoose Schema erstellt. Mit diesem Schema lassen sich die value Felder und dessen Konventionen Definieren. Beispielsweise kann definiert werden welchn Datentyp ein feld hat und welche Vorraussetzungen erfüllt sein müssen, damit ein Eintrag gemacht werden kann. Dies ermöglicht, dass jedem neuen Benutzer automatisch 1000 Geldeinheiten zur verfügung stehen.

### 4.7.6 Controllers

#### 4.7.6.1 AuthController

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte BeschreibungHier werden wesentliche Authentifizierungs- und Autorisierungsfunktionen implementiert. Zuerstwird eine Funktion namens signToken definiert, die zur Erstellung eines JSON Web Tokens (JWT) dient, welcher die Identität eines Benutzers sichert und eine festgelegte Gültigkeitsdauer hat.

Abbildung 8: SignIn Token

In der signup-Funktion können sich neue Benutzer registrieren, indem ihre in der Datenbank gespeichert werden. Nach erfolgreicher Registrierung wird ein JWT erstellt und zusammen mit den Benutzerdaten zurückgesendet, was den erfolgreichen Abschluss des Vorgangs bestätigt.

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte Beschreibung

Abbildung 9: SingUp Funktion

Die login-Funktion ermöglicht es Benutzern, sich einzuloggen, indem sie ihre E-Mail und ihr Passwort überprüft. Nach der Verifizierung dieser Daten wird ebenfalls ein JWT generiert und ausgegeben, welcher für nachfolgende Anfragen zur Authentifizierung verwendet wird.

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte Beschreibung

Abbildung 10: LogIn Function

Ein Bild, das Text, Screenshot enthält.

Automatisch generierte BeschreibungDie protect-Funktion dient als Middleware, die den Zugang zu bestimmten Routen einschränkt. Sie überprüft das Vorhandensein und die Gültigkeit eines JWT im Anfrage-Header, validiert diesen und stellt sicher, dass der zugehörige Benutzer noch existiert und sein Passwort seit der Token-Ausstellung nicht geändert hat. Bei erfolgreicher Überprüfung wird der Benutzer für die nachfolgenden Prozesse autorisiert, andernfalls wird der Zugriff verwehrt.

Abbildung 11: Protection Function

#### 4.7.6.2 UserController

Hier werden verschiedene Funktionen zur Verwaltung von Benutzerdaten bereitgestellt. Die Funktionen interagieren mit einem Mongoose-Modell namens User und bieten eine Reihe von Endpunkten zur Bearbeitung von Benutzerinformationen.

Ein Bild, das Text, Screenshot, Schrift, Software enthält.

Automatisch generierte BeschreibungDie Funktion getAllUsers holt alle Benutzer aus der Datenbank und sendet sie als Antwort zurück. Im Erfolgsfall wird ein Statuscode 200 zurückgegeben, zusammen mit einer Liste aller Benutzer und der Gesamtanzahl. Bei einem Fehler gibt die Funktion einen Statuscode 404 zurück und informiert den Client über das Problem.

Abbildung 12: GetAllUser EndPoint

Die getUser Funktion sucht einen spezifischen Benutzer anhand seiner ID. Wenn der Benutzer gefunden wird, wird er zusammen mit einem Erfolgsstatus zurückgegeben. Wenn kein Benutzer gefunden wird, sendet die Funktion eine 404-Antwort mit einer entsprechenden Fehlermeldung.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 13: GetUser EndPoint

updateUserBalance ermöglicht es, das Guthaben eines Benutzers zu aktualisieren. Die Funktion prüft zunächst, ob die übergebene Balance-Änderung eine gültige Zahl ist. Anschließend wird versucht, den Benutzer in der Datenbank zu finden und dessen Guthaben zu aktualisieren. Bei einem Fehler oder wenn der Benutzer nicht gefunden wird, wird eine entsprechende Fehlermeldung zurückgegeben.

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte Beschreibung

Abbildung 14: UpdateUserBalance EndPoint

Die Funktionen createUser, updateUser und deleteUser sind Platzhalter, die angeben, dass diese Routen noch definiert werden müssen. Sie geben derzeit einen Statuscode 500 zurück und informieren den Nutzer, dass die entsprechende Funktionalität noch nicht implementiert ist.

#### 4.7.6.3 ErrorController

Dies ist ein Fehlerbehandlungsmodul, das verschiedene Arten von Datenbank- und Authentifizierungsfehlern abfängt und entsprechende Benutzerfehlermeldungen generiert.

## 4.8 Frontend

Angular ist ein modernes, umfangreiches Framework für die Entwicklung von Single-Page-Applications (SPAs), das von Google entwickelt und gewartet wird. Es ermöglicht Entwicklern, leistungsstarke und dynamische Webanwendungen durch die Verwendung von TypeScript, einer streng typisierten Version von JavaScript, zu erstellen. Angular nutzt ein Komponenten-basiertes Architekturmodell, das die Wiederverwendung von Code fördert und die Wartung erleichtert.

Hier werden die wichtigsten Elemente aufgelistet, die in dem Projekt verwendet werden.

|  |  |
| --- | --- |
| **Element** | **Beschreibung** |
| Module | Diese bündeln verwandte Codebestandteile zusammen. Ein Modul definiert einen Rahmen, in dem bestimmte Komponenten zusammenarbeiten können |
| Komponenten | Diese sind die Bausteine der Anwendung und kapseln die Logik der Benutzeroberfläche, die Darstellung und das Verhalten in einer Klasse. |
| Services | Dienste sind wiederverwendbare Klassen, die spezifische Funktionen ausführen, wie z.B. Datenabruf von einem Server, und können in Komponenten injiziert werden, um deren Modularität und Wiederverwendbarkeit zu erhöhen. |
| Routing | Das Angular-Router-Modul ermöglicht die Navigation zwischen verschiedenen Ansichten und Komponenten innerhalb einer SPA, ohne die Seite neu laden zu müssen. |

### 4.8.1 App.component.html

Beinhaltet HTML Elemente die auf allen Seiten angezeigt werden. In diesem Projekt wird die Navbar auf jeder Seite angezeigt. Darum wird sie hier implementiert. Zudem beinhaltet diese Seite den Router-Outlet. In Angular, das für die Entwicklung von Single-Page-Applications (SPAs) genutzt wird, wird jede Unterseite als sogenanntes "Outlet" im Router dargestellt. Der Router ist ebenfalls in dieser Struktur enthalten und ermöglicht die dynamische Anzeige verschiedener Ansichten innerhalb der gleichen Seite.

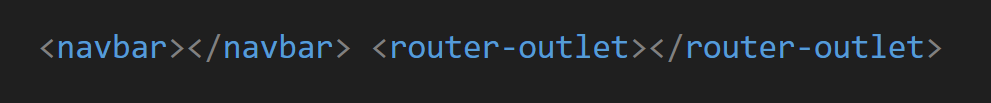


Abbildung 15: AppComponentHtml

### 4.8.2 App Routes

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungIn dieser Datei werden die Routen definiert, die dem Router angeben, was er rendern soll. Dabei wurde für jede Page die geplant wurde eine route erstellt und dem routes-Array hinzugefügt. Dieses hat den tag «export» um in anderen Dateien referenziert werden zu können. Der erste Eintrag im Array ist ein leerer String. Dies bewirkt, dass wenn die Url leer ist, automatisch auf die «welcome» page weitergeleitet wird.

Abbildung 16: Routes Array

### 4.8.3 Components

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

Automatisch generierte BeschreibungDann wurden alle Components, die in diesem Fall die einzelnen Pages darstellen, erstellt und implementiert. Diese wurden im erstellten «components» folder angelegt, um den Code übersichtlich zu halten.

#### 4.8.3.1 Home Component

Dieser dient als Hauptscreen nach dem man sich erfolgreich regestiert und eingeloggt hat. Von hier aus kann man die einzelnen Spiele starten. Zudem werden hier Infos zum angemeldeten User gegeben.

#### 4.8.3.2 Welcome Component

Dieser Screen wird neuen oder nicht angemeldeten Nutzern angezeigt. Dieser beinhaltet simplen Text um die Platform anzupreisen.

#### 4.8.3.3 Navbar Component

Abbildung 17: Components Struktur

Hier werden kann man sich zum Homescreen navigieren oder sich registrieren bzw. an-/abmelden.

#### 4.8.3.4 SignUp Component

In diesem Component wird die Registrierung abgewickelt. Dabei werden die Daten im Formular «signUpForm», welches eine verwendung der FormBuilder-Klasse ist, validiert und gespeichert.

Abbildung 18: SignUpForm

Ein Bild, das Text, Screenshot, Display enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Schrift, Screenshot, Reihe enthält.

Automatisch generierte BeschreibungDieses wird dann «onSubmit» an den Auth Service geschickt. Dort wird es dann an den API-Endpoint (/signup) geschickt, wo es dann vom Server validiert wird. Wenn dies erfolgreich geschehen ist, schickt der Server die daten an die MongoDB Datenbank.

Abbildung 19: SignUp AuthService

#### 4.8.3.5 LogIn Component

Dieser Component beinhaltet die Login Funktionalität und bearbeitet die Daten ähnlich dem SignUp Component. Die vom Benutzer eingegebenen Daten werden gebündelt. Dann werden sie über den AuthService an den (/login) API-Endpoint gesendet. Nach serverseitiger Validierung werden die Daten mit denen der Datenbank eingetragenen verglichen und bei Übereinstimmung authentifiziert.

### 4.8.4 Game Logik

Nach dem die Login- und Registrierungslogik im Backend wie im Frontend implementiert war, wurden die Spiele Blackjack und Slots erstellt.

# Projektplanung

Abbildung 20: Protection Function 2

## User Stories

1. Als Nutzer möchte ich mich mit einem Nutzerkonto registrieren und spielen können.
2. Als Nutzer möchte ich mein Guthaben bei Blackjack und in den Slot Maschinen synchronisiert haben.
3. Ich möchte Das meine Persönlichen Daten sicher gespeichert werden.
4. Als Nutzer möchte ich selber entscheiden mit wie viel Einsatz ich spiele.
5. Als Nutzer möchte ich eine simple und moderne Nutzeroberfläche haben.

## Features

1. -> Loginsystem
2. -> General synchronisiertes Bankkonto in DB
3. -> Verschlüsselung durch JWT Verwendung
4. -> verschieden hohe Einsätze in den Spielen
5. -> Optisch ansprechendes UI/UX durch Angular Material Design

## FlowchartEin Bild, das Text, Screenshot, Schrift, Zahl enthält. Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

## Mockup

## Glossar

|  |  |
| --- | --- |
| **Begriff** | **Definition** |
| Scrum | Agiles Rahmenwerk für iterative Entwicklungen von Softwareprodukten. Arbeit wird in festgelegten Zeiträumen sogenannten Sprints, durchgeführt. |
| Destruktives Testen | Testmethode, die absichtlich Softwarefehler durch extreme oder ungewöhnliche Bedingungen provoziert, um Stabilität zu prüfen. |
| SPAs (Single-Page Applications) | Webanwendungen, die auf einer einzigen HTML-Seite laden und dynamisch Inhalte aktualisieren. Benutzerinteraktionen erfordern kein vollständiges Seitenneuladen, was zu einer flüssigeren Benutzererfahrung führt. |
|  |  |
|  |  |

# Abbildungsverzeichnis

# Quellenverzeichnis