# IPA-Simulation

Raphael Pfeffinger

Mon

## Inhaltsverzeichnis

Inhalt

[IPA-Simulation 1](#_Toc169603621)

[Inhaltsverzeichnis 2](#_Toc169603622)

[1. Projektmanagement 3](#_Toc169603623)

[1.1 Projektmethodik 3](#_Toc169603624)

[1.2 Zeitplanung 4](#_Toc169603625)

[1.3 Entscheiden 5](#_Toc169603626)

[1.4 Testkonzept 5](#_Toc169603627)

[1.4.1 Testziele 5](#_Toc169603628)

[1.4.2 Teststufen 5](#_Toc169603629)

[1.4.3 Testobjekte 6](#_Toc169603630)

[1.4.4 Testfälle 6](#_Toc169603631)

[1.5 Arbeitsjournal 8](#_Toc169603632)

[2. Projektdokumentation 8](#_Toc169603633)

[2.1 Informieren (Analyse) 8](#_Toc169603634)

[2.2 Planen (Design) 10](#_Toc169603635)

[2.3 Realisieren (Implementieren) 10](#_Toc169603636)

[2.3.1 Datenbank erstellen 10](#_Toc169603637)

[2.3.2 Startseite, Login Seite, Registrierseite 10](#_Toc169603638)

[2.3.3 API Methoden Zentral 10](#_Toc169603639)

[2.3.4 Hash Algorithmus 10](#_Toc169603640)

[2.3.5 Token Konfiguration 10](#_Toc169603641)

[2.4 Kontrollieren (Test) 10](#_Toc169603642)

[2.5 Auswerten 10](#_Toc169603643)

# Projektmanagement

## Projektmethodik

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungAls Projektmethode verwende ich das IPERKA Modell. Die klare Struktur dieses Modells hilft bei der Umsetzung des Projekts im Ganzen. Die Informationsbeschaffung wird im 2. Teil des Dokuments klarer beschrieben werden, und ist ebenfalls im Arbeitsjournal ersichtlich. Die Planung im Sinne der Zeit befindet sich im ersten Teil und wird fortgeführt im 2. Teil mit der Planung der Komponenten und Anwendungsstrategien jener Komponenten. Die Restlichen Punkte sind ausschliesslich in der Dokumentation enthalten. IPERKA wird grundsätzlich als Projektmethodik verwendet und auch als Gerüst der Dokumentation. Im Speziellen verwende ich für die Entwicklung das Wasserfallmodell, welches IPERKA sehr ähnelt.

https://konstrukteur-in.ch/wp-content/up 1

## Zeitplanung

Zeitplanung in Form eines Balkendiagramms mit den Schritten des Wasserfallmodells.

Die Tage sind in 4 Abschnitte unterteilt, 2 am Morgen (8:30-10:35 & 10:50-12:15) und 2 am Nachmittag (13:15-15:00 & 15:10-17:30). Im Diagramm sind die einzelnen Schritte von 1-4 bewertet und entsprechen den tagesabschnitten. Die Dokumentation ist immer Inbegriffen, sowie das Arbeitsjournal.

## Testkonzept

### Testziele

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Beschreibung | Messgrösse | Priorität\* |
| 1 | Schnittstellen test erfolgreich | Korrektheit der Daten in der Datenbank oder eingegeben | 1 |
| 2 | Login test erfolgreich | Session aktiviert, andere Startseite | 1 |
| 3 | Testdaten erfolgreich | Falsche daten lassen sich nicht einfügen | 2 |
| 4 | Registrierung erfolgreich | Daten korrekt mit DB ausgetauscht | 1 |

Priorität: 1=hoch, 2=mittel, 3=tief

### Teststufen

Verwendet werden hauptsächlich Integrationstests und wo möglich Unittests. Zuerst werden Unittests in einer Testumgebung durchgeführt, wo möglich. Als nächstes mache Ich Integrationstests mit dem Ziel die korrekte Zusammenarbeit zwischen DB und API zu garantieren.

### Testobjekte

|  |  |  |
| --- | --- | --- |
| Nr. | Objekt | Beschreibung |
| 1 | Login und Registrierungsseite | Die Eingabefelder mit den Restriktionen müssen getestet werden |
| 2 | API | Die REST API muss getestet werden ob alle CRUD-Funktionen die Daten korrekt verarbeiten. |
| 3 | Datenbank | Die Datenbank muss mit Testdaten gefüttert werden |
| 4 | Sicherheitslayer | Der Sicherheitslayer mit dem Erstellen des Cookies und der Session muss getestet werden |

### Testfälle

|  |  |  |  |
| --- | --- | --- | --- |
| ID/Bezeichnung | 001 | Öffnen der Login Seite |  |
| Beschreibung | Die Login Seite muss geöffnet werden und die Felder sollten verfügbar sein. | | |
| Testvoraussetzung | Programm muss gestartet sein, die Startseite muss geöffnet sein | | |
| Testschritte | 1. In die URL-Leiste /Login eingeben (oder auf Login Link klicken) | | |
| Erwartetes Ergebnis | Die Login Seite öffnet sich und der Button ist deaktiviert, da Felder leer. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID/Bezeichnung | 002 | Öffnen der Register Seite |  |
| Beschreibung | Die Registrierungsseite muss garantieren das der Besucher sich einen Account einrichten kann. | | |
| Testvoraussetzung | Programm muss gestartet sein und Startseite geöffnet | | |
| Testschritte | 1. In die URL-Leiste /Registrieren eingeben (oder auf link auf der Startseite klicken) | | |
| Erwartetes Ergebnis | Die Registrierungsseite öffnet sich und die Felder sind deaktiviert und leer. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID/Bezeichnung | 003 | Erstellen von Account und Login |  |
| Beschreibung | Der Benutzer soll einen Account erstellen, um sich danach einloggen zu können | | |
| Testvoraussetzung | Registrierungsseite muss geöffnet sein und Felder sind deaktiviert (Testfall: 002). | | |
| Testschritte | 1. Valide Daten in die Felder Eingeben 2. Auf Den Button Abbrechen klicken 3. Nochmals Daten eingeben 4. Auf Button Speichern klicken 5. Zu Login Seite gehen 6. Mit den gespeicherten Daten Anmelden | | |
| Erwartetes Ergebnis | Nach Klick auf Button Abbrechen müssen die Daten verschwinden. Wenn die Daten korrekt eingegeben werden und speichern gedrückt wird soll zum Start zurückgebracht werden.  Nach Login soll wieder auf Startseite gebracht werden, wo ein grosser Text mit «LOGGED IN» stehen muss. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID/Bezeichnung | 004 | Negativtest von Anmeldedaten |  |
| Beschreibung | Um zu gewährleisten, dass die Restriktionen der Eingebbaren Daten funktionieren, führe ich einen Negativtest durch. | | |
| Testvoraussetzung | Registrierungsseite geöffnet, Felder deaktiviert. | | |
| Testschritte | 1. Als E-Mail keine E-Mail angeben 2. Als Passwort etwas einfaches wie «Passwort» eingeben | | |
| Erwartetes Ergebnis | Das Formular sollte nicht speicherbar sein resp. Es kommt eine Fehlermeldung mit der Meldung, dass die Daten nicht den Voraussetzungen entsprechen. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID/Bezeichnung | 005 | Testen von API CRUD |  |
| Beschreibung | Die API CRUD Funktionen müssen getestet werden. Dafür gibt es einen automatischen Test | | |
| Testvoraussetzung | 1. Postman muss installiert und geöffnet sein 2. Die Postman Collection und Environment müssen in Postman geöffnet sein. | | |
| Testschritte | 1. Nach öffnen der Collection Rechtsklick auf Ordner 2. Dann auf run | | |
| Erwartetes Ergebnis | Es Sollte alle API-Funktionen testen, welche sich in der Collection befinden und überall sollte das Ergebnis 200 OK sein. | | |

## Arbeitsjournal

# Projektdokumentation

## Informieren (Analyse)

Ziel des Projekts ist das Bauen eines Logins. Das Login ist nicht einfach normal, sondern es ist auf Sicherheit bedacht. Das bedeutet, nach dem Bauen der Grundstruktur mit Start-, Login- und Registrierungsseite, werden die grössten Sicherheitsprobleme, welche mit dem Login zusammenhängen ausgemerzt. Ziel ist es zu verstehen, was die grössten Lücken heutzutage sind.

Zu den häufigsten Risiken gehören laut Strongdm.com [[1]](#footnote-1)die folgenden:

* **Brute-Force Attack**: Eine Attacke bei der versucht wird das Passwort mit einer Wörterliste herauszufinden, indem man Anfrage um Anfrage schickt.

**Lösung:** die Login Versuche limitieren, 2-Faktor-Atuhentifizierung und starke Passwörter

* **Schwache Credentials**

**Lösung:** beim Login Passwortstärke vorgeben (z.B. Menge an Zeichen, Sonderzeichen, Zahlen usw.)

* **Poor Session Management:** Session identifiers sind nicht geschützt, Sessions können von Angreifern übernommen werden, Sessions werden nicht zerstör bzw. sind immer valid.

**Lösung:** eindeutige Session identifiers, sicherstellen dass Sessions zerstört und gelöscht werden.

* **Staying logged in & Cookie hijacking:** wenn es einen remember me button gibt, können Angreifer das Erstellungsmuster von Cookies erraten und den Account übernehmen mithilfe von XSS (Cross site scripting) und Brute force Attacks.

**Lösung:** cookie verschlüsseln, den Remember me button weglassen.

* **SQL-Injections:** Methode bei der Mithilfe von SQL-Befehlen im Login bestimmte Accounts übernommen werden.

**Lösung:** Parameter basierte Datenbank, Stored Procedures

## Planen (Design)

In der Zeit halte ich mich an den Zeitplan aus dem 1. Teil des Dokuments. Eine Rollenzuweisung gibt es nicht, da ich allein arbeite.

**Datenbankmodell (ERM):**

**Ein Bild, das Text, Schrift, Screenshot, Zahl enthält.

Automatisch generierte Beschreibung**

**UML Frontend Klassendiagramm:**

**Ein Bild, das Text, Diagramm, Screenshot, Reihe enthält.

Automatisch generierte Beschreibung**

**Ein Bild, das Text, Screenshot, Schrift, Diagramm enthält.

Automatisch generierte BeschreibungUML API Klassendiagramm:**

### Ein Bild, das Text, Screenshot, Schrift, Zahl enthält. Automatisch generierte BeschreibungMockups

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

Automatisch generierte Beschreibung

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

## Entscheiden

Die Applikation wird in ASP.NET Core umgesetzt und in C# geschrieben. Das liegt daran, dass dieses Framework mir bereits bekannt ist, und mit C# kann ich gut Arbeiten, ausserdem geht das Ganze nach MVC-Architektur. Die API ist eine REST-Schnittstelle und hat einen OData Standard. Die Datenbank ist in MSSQL geschrieben und läuft auf SSMS (SQL Server Management Studio).

Server, Datenbank und API laufen jeweils in einem Docker Container.

Für die Versionierung und Sicherung verwende ich GitHub[[2]](#footnote-2).

**Verwendete Packages:**

### Backup Lösung

Wie oben erwähnt wird GitHub verwendet für die Versionierung. Es dient auch zur Sicherheit als Backup. Die Dokumente für die Projektdokumentation befinden sich ebenfalls im Repository und können jederzeit wiederhergestellt werden.

**Beweis:**

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

Automatisch generierte Beschreibung**

**Dokumentation:**

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

Automatisch generierte Beschreibung**

**Projekt:**

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

Automatisch generierte Beschreibung**

### Login Security

Als Security Lösungen für das Login dient die bei kap. 2.3 genannte Liste als Orientierung. Insgesamt werden im ersten Schritt folgende implementiert:

* Stored Procedures
* Password and E-Mail Restrictions (mind. 10 Zeichen mit Zahl und Sonderzeichen)
* Session Management mit automatischer Zerstörung der Session
* Verschlüsseltes Cookie

## Realisieren (Implementieren)

### Datenbank erstellen

Ich habe als ersten Schritt die Datenbank aufgesetzt in einem Container. Es handelt sich um MSSQL. Danach habe ich die Datenbank Datei erstellt und die erste und einzige Tabelle eingefügt.

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

Automatisch generierte Beschreibung

**Probleme:**

Der Datenbank Container wollte nicht exposed werden auf dem Port 1433. Die Lösung dazu war im docker-compose.override.yaml den Port zu definieren.

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

Automatisch generierte Beschreibung

Diese Datei wird von Visual Studio automatisch erzeugt, wenn eine Applikation Containerized wird. Sie wird erstell zu debugg zwecken und wird ebenfalls von Docker-Compose gelesen.

### Startseite, Login Seite, Registrierseite

Zuerst habe ich die Loginseite erstellt. Dabei

### API Methoden Zentral im Frontend

### Hash Algorithmus

### Token Konfiguration

## Kontrollieren (Test)

## Auswerten

1. <https://www.strongdm.com/blog/authentication-vulnerabilities> [↑](#footnote-ref-1)
2. https://github.com/ruffy777/Secure\_Login.git [↑](#footnote-ref-2)