Post It

Modul 151

Zug GIBZ

20.06.2022

Robin Ruzza & Julia Sorrentino

Inhaltsverzeichnis

[Informieren: 4](#_Toc106659935)

[Aufgabenstellung: 4](#_Toc106659936)

[Planen: 5](#_Toc106659937)

[Zieldefinition: 5](#_Toc106659938)

[Meilensteine: 5](#_Toc106659939)

[Entscheiden: 5](#_Toc106659940)

[Realisieren: 5](#_Toc106659941)

[4-Tier Architektur 5](#_Toc106659942)

[Datenbankanbindung/CRUD 6](#_Toc106659943)

[Datensicherheit: 6](#_Toc106659944)

[Angriffsmöglichkeiten 6](#_Toc106659945)

[-HTML Injection 6](#_Toc106659946)

[-SQL Injection 6](#_Toc106659947)

[-Cross Site Scripting 6](#_Toc106659948)

[-Cross Site Request Forgery (CSRF) 6](#_Toc106659949)

[-Session Hijacking 6](#_Toc106659950)

[-Directory Traversal 7](#_Toc106659951)

[-Denial of Service, Distributed Denial of Service (DoS) 7](#_Toc106659952)

[-E-Mail Injection 7](#_Toc106659953)

[-Man in the Middle Attack 7](#_Toc106659954)

[-Phishing 7](#_Toc106659955)

[-Shell Injection 7](#_Toc106659956)

[-Brute-Force 7](#_Toc106659957)

[- usw. 7](#_Toc106659958)

[Ergriffene Massnahmen 7](#_Toc106659959)

[1. Passwort Mindestanforderungen: 7](#_Toc106659960)

[2. Encryptung vom Passwort 7](#_Toc106659961)

[3. E-Mail-Validierung 7](#_Toc106659962)

[4. Https Verschlüsslung / SSL 7](#_Toc106659963)

[Journal : 8](#_Toc106659964)

[Tag 1: 8](#_Toc106659965)

[Tag 2: 8](#_Toc106659966)

[Tag 3: 8](#_Toc106659967)

[Tag 4: 8](#_Toc106659968)

[Tag 5: 9](#_Toc106659969)

[Kontrollieren: 10](#_Toc106659970)

[11](#_Toc106659971)

[Auswerten: 11](#_Toc106659972)

# Informieren:

## Aufgabenstellung:

Für Ihre Applikation erstellen Sie ein GIT Repository. In der Dokumentation halten Sie fest welche Features/Funktionen Teile der Software von wem realisiert werden.

Diese Liste sollte mit Ihren regelmässigen Commits im Software Repo übereinstimmen.

Es wird erwartet das jeder Schüler/jede Schülerin mindestens 5 Gehaltvolle Commits leistet. Ein Commit zählt nur wenn er wesentliche Änderungen enthält.

Gibt es an einem Schultag keine Code-Änderungen dokumentieren Sie das in Ihrer Dokumentation. (Journal Ihrer Arbeiten)

Die Projektarbeit realisieren Sie in einer ausgewählten Technologie. Sie verwenden generelle Techniken wie HTML, CSS und JavaScript. Wenn Sie Frameworks einsetzen, begründen Sie die Wahl in der Dokumentation.

|  |  |
| --- | --- |
| Für die Bewertung der Arbeit werden unter anderem folgende Kriterien beachtet: Formale Anforderungen | Titel, Name, Thema, Datum, Seitenzahl usw. |
| Generelle Informationen | Spezifikation / Zieldefinition / Meilensteine |
| Architektur | Min. 4-Tier Architektur  Datenbankanbindung / CRUD  Datensicherheit wird berücksichtigt (Themen aus «Sicherheit in Web-Applikationen»)  Sessions / Transationen / SSL/TLS |
| Arbeitsjournal | Wer hat wann was gemacht? Es muss ersichtlich sein wer welche Arbeiten gemacht hat. |
| Datensicherheit | Ihre Anwendung ist gegen die gelernten Angriffe geschützt. In der Dokumentation gehen Sie auf die verschiedenen Angriffsmöglichkeiten sowie die ergriffenen Massnahmen ein. |

Sie dokumentieren alle Ihre Entscheide und Arbeiten in einem Dokument. Die Bewertung Ihrer Leistung findet Anhand des Dokuments statt.

Sie präsentieren Ihre Arbeit Ihren Kollegen. Bewertet werden folgende Kriterien

- Die Präsentation startet nach maximal zwei Minuten Vorbereitungszeit

- Die Präsentation dauert maximal 5 Minuten

- Einsatz von Präsentationsmitteln (z.B. Beamer, Tafel, Whiteboard usw.)

- Allgemeine Präsentationsmerkmale (z.B. Blick in Klasse, deutliches Sprechen usw.)

# Planen:

## Zieldefinition:

Unser Hauptziel ist es, eine vollständige und professionelle Dokumentation für unsere Software herzustellen. Dabei ist es wichtig, dass wir unser Journal bei jeglichen Änderungen/Erweiterungen/Updates von unserer Software festhalten und dokumentieren. Wir müssen dabei das Abgabedatum sehr im Auge behalten, da es durch eine verspätete Abgabe eine Note Abzug gibt.

Wir müssen auch jede Woche mindestens ein Commit machen, und darauf achten, dass jeder Schüler mindestens 5 Commits macht. Somit müsste jeder einmal pro Woche ein Commit machen, damit alles aufgeht.

Ein weiterer wichtiger Teil dieses Projektes ist die Datensicherheit. Wir müssen unsere Software so gestalten, dass Angriffe gar nicht, oder nur leicht möglich sind. Die Massnahmen, die wir einführen werden, werden von Julia recherchiert und dokumentiert, und von Robin umgesetzt.

## Meilensteine:

1. Planung und Entscheidung unseres Vorgehens
2. Erstellung des Grundgerüstes der Software
3. Anbindung der Datenbank mit der Webseite
4. Recherchieren der Datensicherheit und Anwendung in unserer Software
5. Testen unserer Software und Abschluss des Projektes
6. Dokumentation mit den letzten Informationen vervollständigen

# Entscheiden:

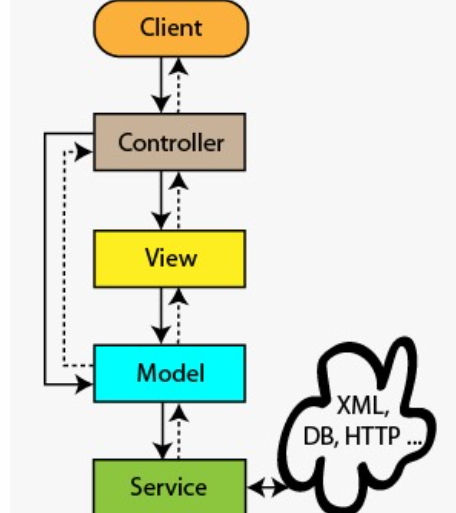
Wir haben uns für die IPERKA Methode entschieden und diese Methode auch angepasst, damit wir ein Journal führen können und die Theorie, die wir benutzt haben, erklären und begründen können.

Wir haben uns für die IPERKA Methode entschieden, da wir beide uns am meisten mit diesem Vorgehen auskennen und da es sehr übersichtlich ist. Bei der Datensicherheit werden wir uns eher auf die Sicherheit der Passwörter widmen. Wir haben uns dafür entschieden, unsere Webseite mit Passwort-Verschlüsselung, und Passwort-Minimal-Anforderungen zu schützen. Wir wollen ebenfalls eine E-Mail-Verifizierung machen; diese soll überprüfen, ob die angegebenen E-Mail Adressen echt ist.

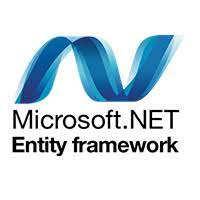
Um die Webseite und deren Daten zu schützen, wollen wir eine Https Verschlüsselung machen. Die Dokumentation und Theorie zu diesen Angriffsmöglichkeiten und die Massnahmen, die wir ergriffen haben, findet man im Abschnitt «Datensicherheit» in unserer Dokumentation.

# Realisieren:

## 4-Tier Architektur

Bei der 4-Tier Architektur (auch 4-Schichten-Architektur genannt) geht es darum, die Prinzipien zur Strukturierung von Software-Architekturen zu definieren. Dabei ist jede Schicht einem bestimmten Aspekt zugeordnet z.Bsp. eine Teil-Funktionalität, eine geschlossene Komponente oder eine Klasse. Das Hauptziel der Schichtenmodelle/Schichtenarchitektur ist es, die Komplexität der Software-Systemen besser zu strukturieren und reduzieren zu können.

## Datenbankanbindung/CRUD

Die Datenbank haben wir über das Microsoft Entity Framework direkt im Code gemacht. Somit mussten wir auch kein SQL schreiben.

Das Entity Framework (auch EF genannt), ist ein Framework für objektrelationale Abbildungen. Wie man auf dem Bild sieht, wurde es von Microsoft entwickelt und dient dem ORM auf .Net-Framework- sowie auf .NET-Objektstrukturen.

Wir haben das Framework verwendet, da es eine einfache Verwaltung der Datenbank direkt aus dem Code ermöglicht, und wir direkt Klassentabellen in der Datenbank erstellen können.

# Datensicherheit:

## Angriffsmöglichkeiten

### -HTML Injection

Durch HTML in der Benutzereingaben, das HTML der Webseite manipulieren.

### -SQL Injection

Durch Metazeichen in der Benutzereingaben, SQL-Aufrufe manipulieren.

### -Cross Site Scripting

Einschleusen von manipulierten Informationen.

### -Cross Site Request Forgery (CSRF)

Eine Cross-Site-Request-Forgery (XSRF/CSRF) ist ein Angriff auf ein Computersystem, bei dem der Angreifer unberechtigt Daten in einer Webanwendung verändert. Er bedient sich dazu eines Opfers, das ein berechtigter Benutzer der Webanwendung sein muss. Dabei muss der Angreifer das Opfer dazu bringen, manipulierte HTTP-Requests auszuführen.

### -Session Hijacking

Der Angreifer muss eine Möglichkeit finden, die URL mit der SessionID des Benutzers auf seinem PC darzustellen. Dies kann durch mitschneiden des Netzwerkverkehrs erfolgen, oder durch das Einschleusen eines einfachen JavaScript-Befehls beziehungsweise HTML Tags, der in einer Webseite auf dem Webserver platziert wird.

### -Directory Traversal

Manipulierte Pfadangaben; und somit Files mit Daten stehlen.

### -Denial of Service, Distributed Denial of Service (DoS)

Vielzahl von Verbindungsanfragen die versuchen, die Ressourcen für die regulären Anfragen einzuschränken beziehungsweise zu entziehen.

### -E-Mail Injection

Der Angreifer fügt manipulierte Daten in ein Kontaktformular ein. Wird meistens für den Versand von SPAM angewendet.

### -Man in the Middle Attack

Der Angreifer versucht sich zwischen Webserver und Benutzer einzuklinken und dabei mit dem Benutzer eine Verbindung aufzubauen ohne dass dieser das merkt. Zuerst muss einem Benutzer eine gefälschte URL untergejubelt werden, z.B. via Mail oder Link in einem Forum/Blog.

### -Phishing

Meist massenweise per E-Mail gesendete Aufforderungen, Zugangsdaten für z.B.: Online-Banking etc. auf einer Webseite einzugeben. Die Seite sieht äusserlich wie die Webseite des Betreibers aus, unterliegt aber der Kontrolle des Angreifers.

### -Shell Injection

Ausführen von Konsolenbefehlen auf dem Server über Missbrauch eines ungesicherten oder selber eingeschleusten "shell()"-Befehls (PHP).

### -Brute-Force

Einfache Passwörter können durch Brute-Force-Attacken, also durch zufälliges Ausprobieren (auch durch automatisches Pröbeln durch Roboter mit entsprechenden Wörterbüchern von häufig verwendeten Passwörtern), geknackt werden.

### - usw.

## Ergriffene Massnahmen

### Passwort Mindestanforderungen:

Um zu verhindern, dass eine Person das Passwort unserem User «errät, haben wir eine Mindestanforderung für das Passwort erstellt. Dabei muss das Passwort mindestens 8 Zeichen lang sein, das Passwort sollte mindestens ein kleiner Buchstabe und ein grosser Buchstabe enthalten. Ebenfalls ist mindestens ein Sonderzeichen erforderlich.

### Encryptung vom Passwort

Passwort wird nicht im Klartext abgelegt, sondern wird mit Hilfe eines Hashes verschlüsselt, bevor es in die Datenbank gespeichert wird.

### E-Mail-Validierung

Da es Eventuell-Benutzer gibt, die eine falsche oder nicht-existierende E-Mail-Adresse angeben, haben wir eine Basic E-Mail Validierung hinzugefügt. Dabei wir die E-Mail-Adresse erst akzeptiert, wenn es dieses Format besitzt: [variable@variable.TLD](mailto:variable@variable.TLD) (Das TLD steht für Top Level Domain).

### Https Verschlüsslung / SSL

Unsere Webseite ist durch HTTPS mit SSL verschlüsselt. Und somit gegen einige der oben genannten Attacken geschützt.

# Journal :

## Tag 1:

Als erstes haben wir uns zusammengesetzt, um zu entschieden, welches Vorgehen wir benutzen wollen. Wir haben uns für die IPERKA Methode entschieden. Danach haben wir die ersten kleineren Aufgaben aufgeteilt. Julia wird grösstenteils die Dokumentation und die Theorie übernehmen, während Robin sich um die Software kümmert. Als erstes fing Julia mit dem Grundgerüst der Dokumentation (IPERKA) an, und füllte dies mit den Anfangsinformationen (das Informieren, das Planen und die Zieldefinition und der erste Tag des Journals).

## Tag 2:

Heute hat Julia mit der Dokumentation angefangen und hat sich mit dem zweiten Teil der Planung befasst, und hat somit die Meilensteine erstellt und diese mit Robin besprochen und wo nötig angepasst. Somit haben wir nun 6 Meilensteine definiert. Nachdem der Planungsteil beendet war, und wir unser Ziel und Meilensteine wussten, hat Julia mit dem Testprotokoll angefangen, damit wir am Ende des Projektes die Ziele und Meilensteine testen können. Wir haben besprochen, ob wir nur ein Testprotokoll für die Software machen wollen, oder ob wir zwei Testprotokolle machen wollen. Eines für die Software inkl. der Datensicherheit und Datenbank, und ein Testprotokoll für die allgemeine Organisation, Administration und Dokumentation.

Wir haben uns schlussendlich für ein einziges Testprotokoll entschieden, darin sollte die Software, die Datenbank und die Datensicherheit getestet werden.

Somit hat Julia nach der Besprechung mit dem Testprotokoll angefangen, und hat zuerst das Grundgerüst für das Testprotokoll erstellt. Dies hat sie danach auch zusammen mit der Dokumentation auf Github hochgeladen. Ihr ist dabei aufgefallen, dass sie für das Committen von Files noch Hilfe braucht, und somit schaute Sie auf [YouTube](https://www.youtube.com/watch?v=iv8rSLsi1xo) ein kurzes Tutorial an.

Robin hat in der Zwischenzeit das Projekt erstellt. Danach hat er das erste Identity Model erstellt. Als nächstes fing er mit den Views für die Login und der Registration an. Dabei hat er natürlicherweise mit Visual Studio gearbeitet, und mit C# und Razor programmiert.

Der Code wurde kommentiert, damit es einfacher nachvollziehbar ist.

## Tag 3:

Da hatten wir beide frei und haben somit nicht an diesem Projekt gearbeitet.

## Tag 4:

Heute hat Julia als erstes mit dem Testprotokoll angefangen und hat diesen ausgefüllt. Am Ende besprach sie noch, ob Testfälle fehlten oder ob man etwas bestimmtes anpassen müsste. Die meisten Ziele/Funktionen, die von uns getestet wurden, sind die Allgemeinen Anforderungen und ein paar Sicherheitsmassnahmen, die wir zusammen entschieden haben. Das Testprotokoll befindet sich im Abschnitt «Kontrollieren» unserer Dokumentation. Das File der Testfälle findet man auch in unserem GitHub unter dem «Dokumentation» Folder.

Beim Lauschen eines Gespräches von anderen Klassenkameraden ist bei uns der Zweifel im Zusammenhang mit den Commits aufgekommen. Diese meinten, dass die Commits nur codebasiert sind, und dass somit jede Person je 5 Commits im Code machen musste. Um dies zu klären, haben wir nachgefragt und es stellte sich heraus, dass z.Bsp. Änderungen und der Dokumentation oder Erstellen eines Testddokumentes nicht zählen würde. Diese Information stellte unsere Planung sehr auf den Kopf, da wir geplant hatten, dass sich Julia mit der Theorie, der Dokumentation und den Testfällen befassen und sich Robin um die Software kümmern würde.

Da die Dokumentation somit nicht zu den Commits zählten, haben wir diese zusammen mit den Testfällen aus dem GitHub entfernt. Wir werden diese erst am Ende als Endprodukt hochladen, da das ständige Aktualisieren und Pushen der Dokumente auch Zeit kostet.

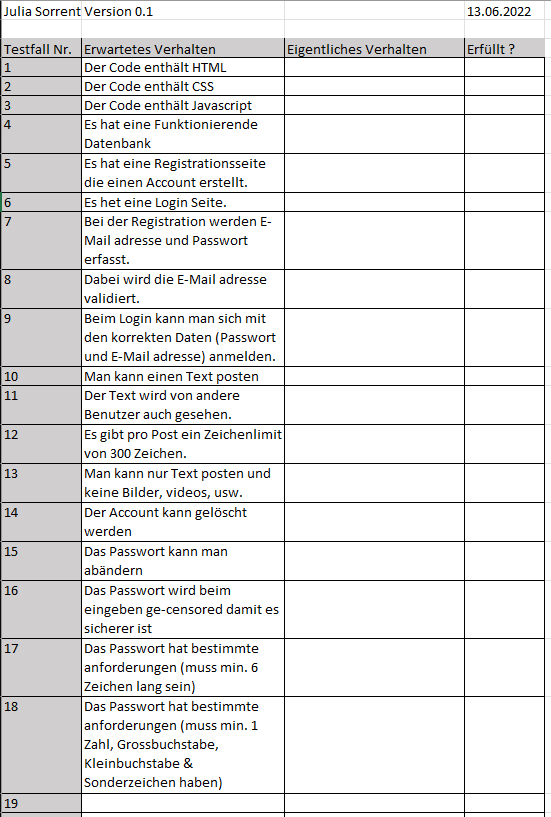
Damit Julia auch mindestens 5 Commits hat, hat Robin ihr einen Auftrag gegeben. Dabei musste sie die Post Klasse erstellen, damit das Posten funktionieren kann. Um dies zu tun, brauchte Julia erstmals noch den .NET Framework 4.8. Dies musste sie installieren und updaten, was lange gebraucht hat. Doch schlussendlich funktionierte es, und sie konnte mit dem Programmieren anfangen. Danach pushte sie den Code auf GitHub.

In der Zwischenzeit hat Robin angefangen, die Differenzierung der Projekt-Architektur aufzubauen. Als er fertig war, hat er noch mit dem Datahandling begonnen und das Model für den Post erstellt. Der Code sollte selbstverständlich sein, wenn man die Kommentare beachtet hat.

## Tag 5:

Heute hat Julia noch die letzten Teile der Dokumentation gemacht, dazu gehören der Teil der Datensicherheit, die Datenbankanbindung, das Kontrollieren/Testen der Webseite und das Auswerten des gesamten Projektes. Bei der Datensicherheit war uns unklar, ob wir alle möglichen (uns bekannten) Angriffsmöglichkeiten recherchieren mussten, und so fragten wir nach. Es stellte sich heraus, dass wir alle möglichen Angriffsmöglichkeiten beschreiben/erklären müssen. Beim Testen lief alles gut und wir konnten alle Testfälle durchführen. Das ausgefüllte Testprotokoll befindet sich im Abschnitt «Kontrollieren» unserer Dokumentation. Robin hat heute mit Julia noch die Webseite finalisiert und alles ausführlich kommentiert und letzten Einträge in die Dokumentation gemacht. Am Schluss, wurde die Dokumentation und das Test File mit dem Code auf GitHub hochgeladen.

# Kontrollieren:

Nachfolgend sieht man unser noch nicht ausgefüllte Protokoll. Wie man sieht, sind die meisten Testfälle über die Login und Registrationsseite. Als Datensicherheit werden wir die Passwörter und die E-Mail-Verifizierung testen.

Auf der nächsten Seite sieht man unser ausgefülltes Testing-Dokument. Robin hat die Tests durchgeführt und das Dokument ausgefüllt. Wie man sieht, haben alle Testfälle genau so funktioniert, wie wir uns das vorgestellt haben. Und somit konnten wir das Testing beenden.

# 

# Auswerten:

Rückblickend hat das Meiste recht gut funktioniert, die Dokumentation und die Software haben wir fertig bekommen. Die Zeit für den Auftrag war unserer Meinung nach recht knapp, da man eine vollständige Planung, Dokumentation mit Journal, Testfälle, eine Datenbank und das Coden in wenigen Wochen machen musste.

Wir haben einige Schwierigkeiten festgestell; diese waren aber auf Missverständnisse zurückzuführen. Eines der grössten Missverständnisse war der Commit Teil; dieses Missverständnis setzte unser Projekt recht auf den Kopf. Wir hatten geplant, die Theorie, Dokumente, den administrativen Anteil und das Coden aufzuteilen und dies auch so bis zur vorletzten Woche weitergeführt. Danach war es aber doch sehr schwierig, noch am Ende einen Weg zu finden, um für Julia 5 sinnvolle Commits im Code zu machen.

Die Datensicherheit in unserer Webseite einzufügen war auch recht schwer, da wir uns in der vorherigen Aufgabe mit dem Social Engineering auseinandergesetzt hatten und dieses Wissen nicht wirklich für unsere Webseite benutzen konnten. Jedoch achteten wir darauf, die Basic Datensicherheitsmassnahmen für E-Mail und Passwort einzuhalten.