

Health-Keeper Handout

Inhaltsverzeichnis

Inhaltsverzeichnis	1
Aufwandsstatistiken	2
Highlights der Demo	5
Highlights des Projekts	8
Architektur	8
Software Tools/Plattform/Technik/Libraries	9
Datenbank Design.....	10
Testing.....	10
Metriken.....	11
Maintainability Index - 86 von 100	11
Cyclomatic Complexity - 242.....	11
Class Coupling Score - 125	11
CI/CD.....	11

Aufwandsstatistiken

	10.10.2023-17.10.2023		17.10.2023-24.10.2023		24.10.2023-31.10.2023	
Person	Aufgabe	Stunden	Aufgabe	Stunden	Aufgabe	Stunden
Christiane	1) Teamsetup 2) Entscheidung über Projektidee und Tech-Stack 4) Einen GitHub Projekt erstellen	3	1) Jira-Projekt einrichten und alle Teammitglieder einladen 2) GitHub Projekt erstellen und alle Teammitglieder einladen 3) Blog einrichten	3	1) SRS Kapitel 1 schreiben 2) SRS Kapitel 4 schreiben 3) Dotnet Grundgerüst aufsetzen	5
Ekaterina		3	1) 5 User Stories schreiben	1	1) SRS Kapitel 2 schreiben 2) Blogeintrag schreiben 3) Kommentare zu anderen Blogbeiträgen schreiben	3
Lukas M.		3	1) Kommentare zu anderen Blogbeiträgen schreiben	1	1) SRS Kapitel 3.1 - 3.3	3
Lukas H.		3	1) Blogbeitrag schreiben	1	1) SRS Kapitel 3.4 - 3.6	3

	31.10.2023-07.11.2023		07.11.2023-14.11.2023		14.11.2023-21.11.2023	
Person	Aufgabe	Stunden	Aufgabe	Stunden	Aufgabe	Stunden
Christiane					1) Controller für die Statistic Seite erstellen 2) JS-Code für BMI-Rechner schreiben	5
Ekaterina	1) Frontend-Mockup erstellen (Homepage) 2) Kommentare zu anderen Blogbeiträgen schreiben	3	1) UML-Klassendiagramm erstellen 2) Blogbeitrag schreiben	4	3) Frontend für Statistic Seite erstellen	3

Lukas M.					1) Kommentare zu anderen Blogbeiträgen schreiben	1
Lukas H.	1) Aktivitäts- und Sequenzdiagramme erstellen 2) Blogbeitrag schreiben 3) UCRS für Login und Passwort-Reset erstellen	4	1) Kommentare zu anderen Blogbeiträgen schreiben	1	1) Blogbeitrag schreiben	2

	21.11.2023-28.11.2021		28.11.2023-05.12.2023		05.12.2023-12.12.2023	
Person	Aufgabe	Stunden	Aufgabe	Stunden	Aufgabe	Stunden
Christiane	1) Controller für alle Seiten erstellen	3	1) Model für Statistic Seite erstellen 2) User Model erstellen	3	1) User Update Model erstellen 2) FoodJournal Model erstellen	3
Ekaterina	1) Frontend für Login- und Registrieren-Seite erstellen	4	1) Utility Tree erstellen 2) Architekturentscheidungen beschreiben 3) Einen Blogpost erstellen	3	1) Software Architecture Abschnitt 2)	1
Lukas M.	1) Grundgerüst für den Food-Journal erstellen 2) Kommentare zu anderen Blogbeiträgen schreiben	4	1) Kommentare zu anderen Blogbeiträgen schreiben	1	1) Food Journal erweitern und Funktionalitäten einfügen	7
Lukas H.	1) Blogbeitrag schreiben	1				

	12.12.2023-19.12.2023		03.04.2024-10.04.2024		10.04.2024-17.04.2024	
Person	Aufgabe	Stunden	Aufgabe	Stunden	Aufgabe	Stunden
Christiane	1) Präsentation vorbereiten	1				
Ekaterina	1) Handout erstellen 2) Frontend für Kalender Seite erstellen	4	1) Login/Register Controller angepasst 2) Einarbeitung in ASP.NET (Backend)	4	1) Calendar front-end anpassen 2) Einarbeitung in ASP.NET (Backend)	3

	3) Präsentation vorbereiten					
Lukas M.	1) Präsentation vorbereiten	1				
Lukas H.	1) Präsentation vorbereiten	1			1) Blogbeitrag schreiben	1
	17.04.2024- 24.04.2024		24.04.2024-08.05.2024		08.05.2024-15.05.2024	
Person	Aufgabe	Stunden	Aufgabe	Stunden	Aufgabe	Stunden
Ekaterina	2) Einarbeitung in ASP.NET (Backend)	2				
Lukas M.	1) FoodJournal Refactoring 2) Blogbeitrag schreiben	3				
Lukas H.	1) Backend Entwicklung	2	1) Backend Entwicklung 2) Blogbeitrag schreiben	3	1) Backend Entwicklung 2) Blogbeitrag schreiben	1

	15.05.2024-22.05.2024		22.05.2024-29.05.2024		29.05.2024-05.06.2024	
Person	Aufgabe	Stunden	Aufgabe	Stunden	Aufgabe	Stunden
Ekaterina	1) CSS anpassen	2				
Lukas M.	1) Einarbeitung in ASP.NET (Backend)	3	1) Einarbeitung in ASP.NET (Backend)	3	1) Blogbeitrag schreiben 2) Technical Review	4
Lukas H.	1) Controller Management angepasst 2) Code Refactoring 2) Blogbeitrag schreiben	4	1) Blogbeitrag schreiben	2	1) Backend Entwicklung	3

	05.06.2024-12.06.2024		Gesamte Stundenanzahl
Person	Aufgabe	Stunden	
Ekaterina	1) Vorbereitung für Präsentation	4	43
Lukas M.	1) CSS und weitere Kleinigkeiten angepasst 2) Vorbereitung für Präsentation	3	40

Lukas H.	1) BMI-Rechner angepasst 2) Backend Entwicklung 3) Vorbereitung für Präsentation	4	36
----------	---	---	----

Highlights der Demo

1. Health Keeper bietet eine Login-Funktion, die es den Benutzern ermöglicht, personalisierte Konten zu erstellen, sich anzumelden und auf ihre individuellen Daten zuzugreifen. Diese Funktion umfasst die Registrierung neuer Benutzer und die Authentifizierung bestehender Konten. Durch die Konto-Funktionalität können Benutzer ihre Gesundheits- und Fitnessdaten sicher speichern und verfolgen.

The screenshot shows the login interface of the HealthKeeper application. The header is purple and contains the app name and navigation links. The login card is centered and contains the following elements:

- Title:** Login
- Nutzername:** Input field containing 'lukas'
- Passwort:** Input field with masked characters (dots)
- Login Button:** A blue button labeled 'Login'
- Link:** 'Erstelle einen Account.'

Abbildung 1: Login Seite

2. Health Keeper verfügt über eine Tabelle zur Gewichtsverfolgung, die es den Benutzern ermöglicht, ihren Gewichtsverlauf visuell zu überwachen und Trends im Laufe der Zeit


zu erkennen. Diese Funktion unterstützt die Nutzer dabei, ihre Fitnessziele zu verfolgen und motiviert zu bleiben, indem sie Fortschritte und Veränderungen anschaulich darstellt. Es gibt auch einen BMI-Rechner, mit dem Benutzer ihren Body-Mass-Index basierend auf Größe und Gewicht berechnen können.

HealthKeeper

Statistiken

Kalender

Food Journal



Eintrag erstellen:

Größe (cm):

Gewicht (kg):

Eintrag hinzufügen

Datum	Gewicht (kg)	Größe (cm)	BMI	Kategorie
11.06.2024 09:16:47	80	123	52,88	Adipositas Grad III
11.06.2024 09:17:09	123	80	192,19	Adipositas Grad III
11.06.2024 17:11:35	90	190	24,93	Normalgewicht

BMI: 25 -

Normalgewicht

Abbildung 2: BMI-Statistiken

- Health Keeper enthält einen Kalender, der es den Benutzern ermöglicht, eigene Termine und Trainingspläne einzutragen und zu verwalten. Diese Funktion hilft den Nutzern, ihre Zeit effizient zu organisieren und sicherzustellen, dass sie ihre Fitness- und Gesundheitsaktivitäten planen und einhalten können.



Abbildung 3: Kalender

4. Health Keeper verfügt über ein Ernährungstagebuch, in das Benutzer Lebensmittel mit ihren Nährstoffen eintragen können, sortiert nach Mahlzeiten. Diese Funktion ermöglicht es den Benutzern, ihre tägliche Nahrungsaufnahme zu verfolgen und sicherzustellen, dass sie eine ausgewogene Ernährung einhalten. Zusätzlich können Benutzer die Einträge aus vorherigen Tagen einsehen, um ihre Nährstoffzufuhr über einen längeren Zeitraum zu überwachen und zu analysieren.

HealthKeeper

Statistiken

Kalender

Food Journal

Mein Food Journal

< Heute >

> Nahrungsmittel hinzufügen

Frühstück

Frühstück hinzufügen

Mittagessen

Mittagessen hinzufügen

Abendessen

Abendessen hinzufügen

Snacks

Snacks hinzufügen

Abbildung 4: Food Journal

Highlights des Projekts

Architektur

Health Keeper als Multipage-Webanwendung wurde, entworfen, um ein breites Spektrum an Gesundheits- und Fitnessfunktionen zu bieten. Die Architektur folgt einem Client-Server-Modell, wobei ASP.NET als Grundlage dient. Dies ermöglicht die Integration verschiedener Technologien wie Entity Framework Core, JavaScript sowie HTML/CSS für ein robustes

Frontend. Die Anwendung ist modular gestaltet, wobei jede Funktion über eigene Controller navigiert wird. Diese beinhalten Grafik zur Gewichtsverfolgung, einen BMI-Rechner, ein Ernährungstagebuch, sowie einen Kalender für Trainingspläne und persönliche Termine. Health Keeper umfasst auch eine Konto-Funktionalität, die es den Benutzern ermöglicht, personalisierte Konten zu erstellen und sich anzumelden. RESTful APIs werden genutzt, um eine klare Kommunikation zwischen den Controllern und dem Server zu gewährleisten. Entity Framework Core erleichtert den Zugriff auf die Datenbank, während GitHub Actions für automatisierte Tests, Builds und Bereitstellungen eingesetzt werden. Das Projektmanagement erfolgt über JIRA, um die Organisation von Aufgaben und Sprints zu ermöglichen. Es werden bewährte Entwurfsmuster wie das MVC-Pattern, das Repository Pattern und Dependency-Injection angewendet, um eine skalierbare und wartbare Anwendung zu gewährleisten.

Software Tools/Plattform/Technik/Libraries

- **C#:** Eine objektorientierte Programmiersprache, die häufig für die Entwicklung von Anwendungen im .NET-Framework verwendet wird.
- **ASP.NET MVC:** Ein Framework für die Entwicklung von Webanwendungen in der .NET-Plattform, das das Model-View-Controller (MVC)-Muster verwendet.
- **EntityFramework Core:** Ein objektrelationales Mapping-Framework für .NET, das die Interaktion mit Datenbanken durch objektorientierte Konzepte erleichtert.
- **NUnit:** Ein Unit-Testing-Framework für die .NET-Plattform, das es Entwicklern ermöglicht, automatisierte Tests für ihre Anwendungen zu schreiben.
- **GitHub Actions:** Eine Funktion von GitHub, die es Entwicklern ermöglicht, automatisierte Workflows für die Erstellung, Tests und Bereitstellung von Software direkt in ihren GitHub-Repositories zu definieren.
- **Jira:** Ein von Atlassian entwickeltes Projektmanagement-Tool, das Teams dabei unterstützt, ihre Arbeit zu planen, zu verfolgen und zu verwalten, indem es Funktionen wie Aufgabenverfolgung, Berichterstattung und Zusammenarbeit bietet.

Datenbank Design

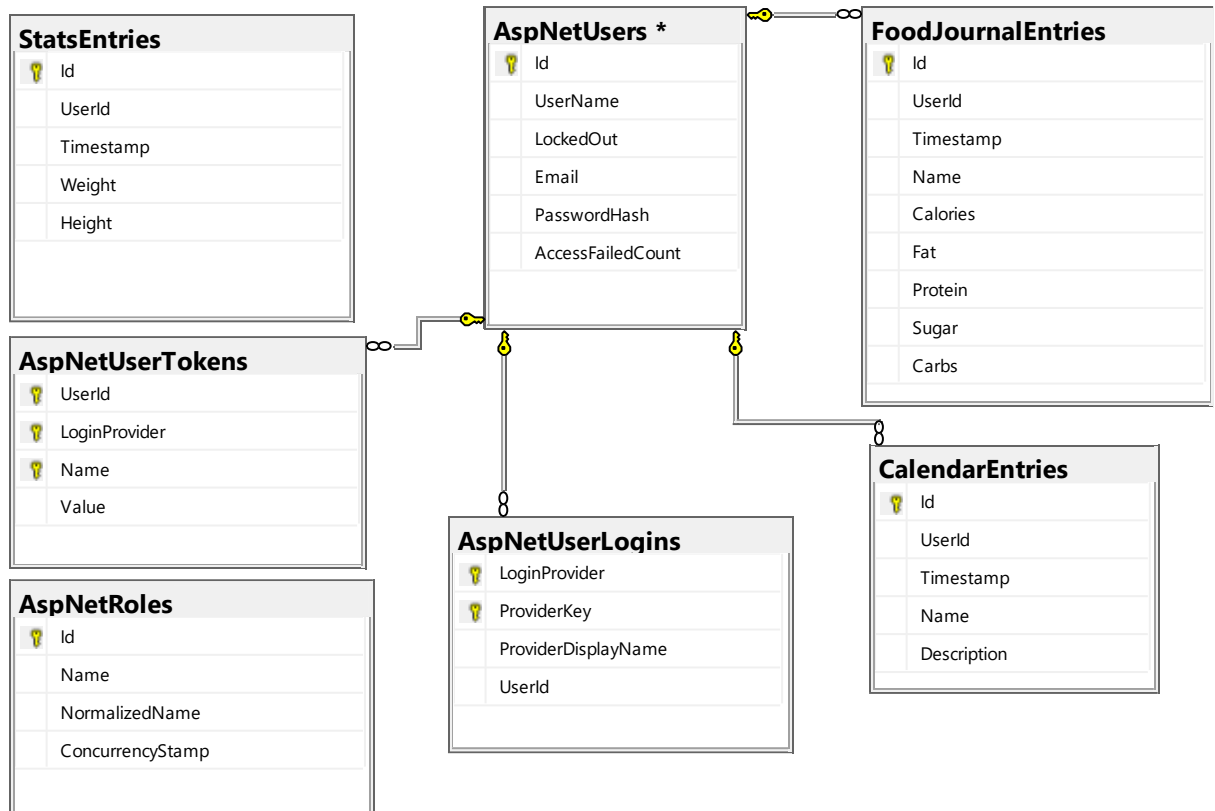


Abbildung 5: ER-Diagramm der Datenbank

Als Datenbank verwenden wir MSSQL mit dem EntityFramework. Das EntityFramework übernimmt für uns die vollständige Datenbank Verwaltung. Die Datenbank Tabellen werden aus C# Klassen generiert und können über automatisch generierte Migrationen geupdatet werden.

Testing

Zum Testen der Anwendung verwenden wir NUnit mit Moq. NUnit ist ein, wie der Name vermuten lässt, Unit Testing Framework für C#. Die Tests liegen hierbei in dem getrennten „HealthKeeper.Tests“ Projekt. Tests werden wie in JUnit mit der [Test] Annotation gekennzeichnet. Moq ist eine Bibliothek, die es erlaubt Klassen zu mocken. Die Code Coverage wird in der CI-Pipeline der Anwendung im Cobertura Format generiert. Eine kurze Zusammenfassung der Ergebnisse wird auf GitHub gepostet.

Aktuell hat das Projekt eine Testabdeckung von etwa 60%. Da besonders UI-Tests eine große Herausforderung darstellen wird darauf verzichtet.

build summary				
Code Coverage 62%				
Package	Line Rate	Branch Rate	Complexity	Health
HealthKeeper.Tests	100%	100%	13	✓
HealthKeeper	39%	54%	145	✗
Summary	62% (223 / 360)	54% (26 / 48)	158	—
Job summary generated at run-time				

Abbildung 6: Code Coverage Report in GitHub Actions

Metriken

Maintainability Index - 86 von 100

Unser System erreicht einen Maintainability Index von 86 Punkten auf einer Skala von maximal 100. Dieser hohe Wert deutet darauf hin, dass unser Code sehr gut wartbar ist. Ein hoher Maintainability Index erleichtert es Entwicklern, Änderungen und Erweiterungen vorzunehmen, ohne unbeabsichtigte Fehler einzuführen. Dies spart nicht nur Zeit, sondern reduziert auch die langfristigen Wartungskosten.

Cyclomatic Complexity - 242

Die Cyclomatic Complexity unseres Systems liegt bei 242. Diese Metrik misst die Anzahl der linearen unabhängigen Pfade durch den Code. Ein höherer Wert deutet auf eine höhere Komplexität hin, was die Testbarkeit und das Verständnis des Codes erschwert.

Class Coupling Score - 125

Der Class Coupling Score unseres Systems beträgt 125. Class Coupling misst, wie stark die Klassen in einem System miteinander verbunden sind. Ein niedrigerer Wert ist wünschenswert, da ein geringer Coupling-Grad die Flexibilität und Wiederverwendbarkeit des Codes erhöht. Ein Wert von 125 ist moderat.

CI/CD

Das Projekt wird bei jedem Push auf das GitHub Repository von einer Continuous Integration Pipeline kompiliert und getestet. Wir nutzen hierfür GitHub Actions, da es einfach in der Benutzung ist. Unsere Pipeline besitzt nur einen einzigen Job, der sich um alle Aufgaben kümmert. Das Berechnen der Code Coverage findet auch in der CI-Pipeline statt.

Auf Continuous Deployment verzichten wir, da es keine Produktionsumgebung gibt, auf welche wir unsere Anwendung aufspielen könnten. Die fertig kompilierte Anwendung kann von GitHub Actions heruntergeladen werden.

build

succeeded 1 hour ago in 39s

- > ✓ Set up job
- > ✓ Pull ghcr.io/irongut/codecoveragesummary:v1.3.0
- > ✓ Run actions/checkout@v4
- > ✓ Setup .NET
- > ✓ Restore dependencies
- > ✓ Build
- > ✓ Test
- > ✓ Code Coverage Summary Report
- > ✓ Write to Job Summary
- > ✓ Upload a Build Artifact
- > ✓ Post Setup .NET
- > ✓ Post Run actions/checkout@v4
- > ✓ Complete job

Abbildung 7: GitHub Actions Pipeline