Ticketsystem

BITLC | Tom Selig
19. Juni 2023

Das Projekt

- Es soll ein Ticketsystem erstellt werden:

  Tester können:

  - Tickets zu Projekten erstellen
  - Tickets den Entwicklern zuweisen

  Entwickler können:

  - Tickets als erledigt markieren

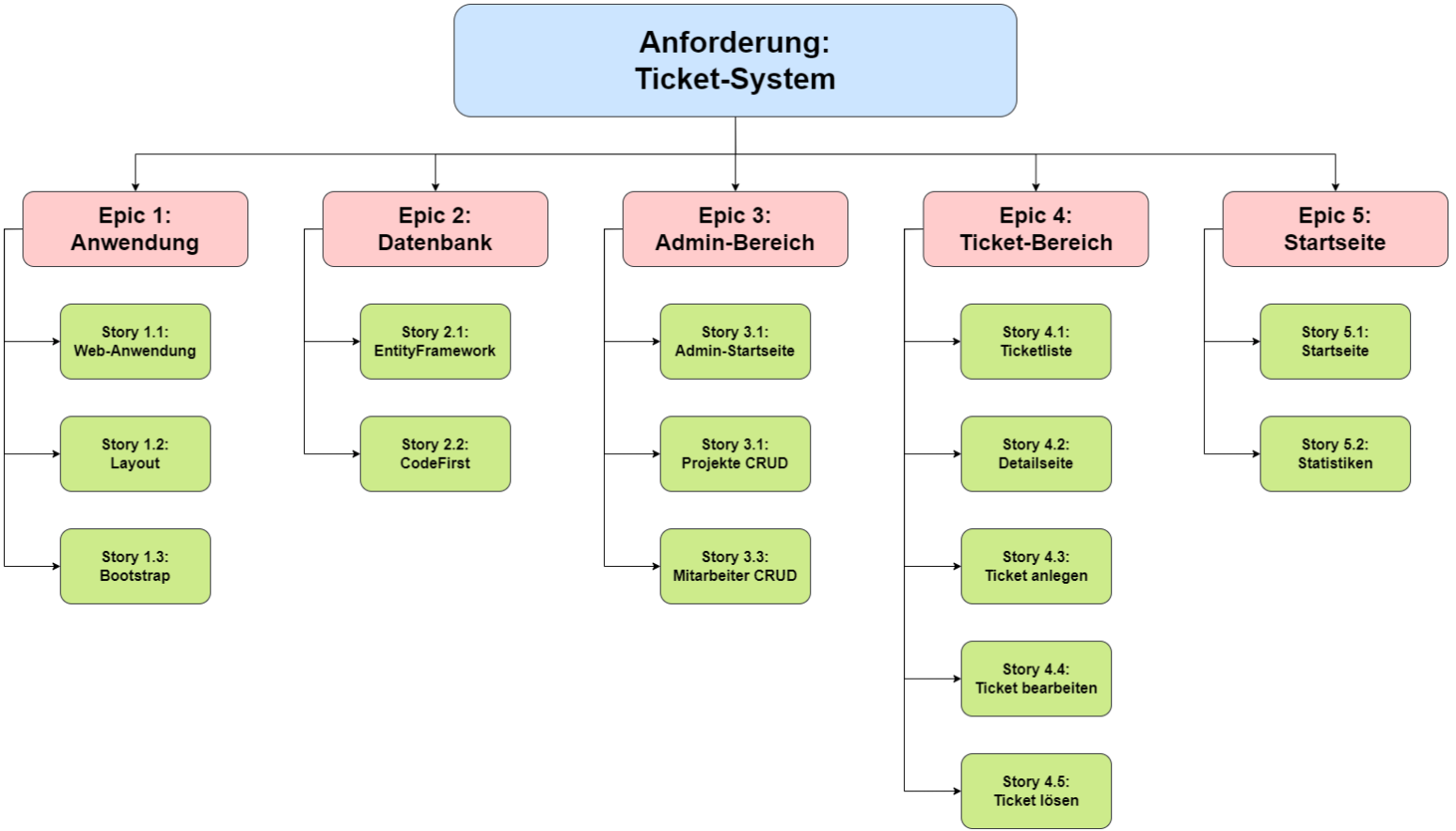
  Admins können:

  - Projekte bearbeiten
  - Mitarbeiter bearbeiten

Der Ablauf

- Das Projekt muss beantragt werden:
  - Projekt-Antrag durch das Projekt-Team
  - Freigabe durch den Dozenten
- Das Projekt soll nach Freigabe des Antrags mit Elementen aus dem Scrum-Framework umgesetzt werden:
  - User-Stories
  - Daily Standups
  - „Sprint“ Planning
  - Reviews

Die Aufgabe


Anforderung:
Ticket-System
Epic 1:
Anwendung
Epic 2:
Datenbank
Epic 3:
Admin-Bereich
Epic 4:
Ticket-Bereich
Epic 5:
Startseite
Story 1.1:
Web-Anwendung
Story 1.2:
Layout
Story 1.3:
Bootstrap
Story 2.1:
EntityFramework
Story 2.2:
CodeFirst
Story 3.1:
Admin-Startseite
Story 3.1:
Projekte CRUD
Story 3.3:
Mitarbeiter CRUD
Story 4.1:
Ticketliste
Story 4.2:
Detailseite
Story 4.3:
Ticket anlegen
Story 4.4:
Ticket bearbeiten
Story 4.5:
Ticket lösen
Story 5.1:
Startseite
Story 5.2:
Statistiken

User-Stories

Epic: #1 - Anwendung

Titel: 1.1 - Web-Anwendung

Story:

Als Benutzer möchte ich, dass das Ticketsystem als Web-Anwendung umgesetzt wird, damit ich von überall aus darauf zugreifen kann.

Akzeptanzkriterien:

- Die Anwendung wird als ASP.NET Core-Projekt umgesetzt.
- Die Anwendung wird nach dem MVC-Muster umgesetzt.
- Wo es sinnvoll ist, kann mit Scaffolding gearbeitet werden.

Notizen des Teams:

-ASP.Net :Leer
-Aktuall Framework

-MVC Muster erzeugen

-Routing im Programm einsetzen
-StaticFiles im Programm einsetzen
-Add(ControllerView)

User-Stories

Epic: #1 - Anwendung

Titel: 1.2 - Layout

Story:

Als Benutzer möchte ich, dass die Anwendung mit einem einheitlichen Layout umgesetzt wird, damit alle aufgerufenen Seiten das gleiche Erscheinungsbild haben.

Akzeptanzkriterien:

- Es wird eine _Layout.cshtml-Datei verwendet.
- Das Layout umfasst einen Header und einen Footer, sowie einen zentralen Bereich für die Anzeige der einzelnen Views.

Notizen des Teams:

HTML5 Version

<Head>
-lang='de'
-StyleSheet.css und link

<Body>
<Header>
-navbar + Link
<Main>
@RenderBody()
<Footer>
-Container + Copyright

User-Stories

Epic: #1 - Anwendung

Titel: 1.3 - Bootstrap

Story:

Als Benutzer möchte ich, dass die Web-Anwendung mit Bootstrap-CSS umgesetzt wird, damit sie das gleiche Erscheinungsbild hat wie die anderen Anwendungen des Unternehmens.

Akzeptanzkriterien:

- Bootstrap ist in allen Views verfügbar.
- Die verwendete Bootstrap-Version bietet Klassen zur Erstellung von Cards.
- Bootstrap wird auf dem eigenen Server gehostet.

Notizen des Teams:

bootstraps.stylesheet für css Version 5.3.0

unter wwwroot Ordner

-im Layout den Bootstrap linken

User-Stories

Epic: #2 - Datenbank

Titel: 2.1 - EntityFramework

Story:

Als Entwickler möchte ich, dass die Web-Anwendung das EntityFramework Core nutzt, damit ich leicht und effizient mit einer SQL Server-Datenbank arbeiten kann.

Akzeptanzkriterien:

- Als Datenbank dient der LocalDB-Server.
- Die NuGet-Pakete für EntityFramework Core passen mit ihrer Version zum ASP.NET Core-Projekt.
- Der ConnectionString wird in der appsettings.json-Datei hinterlegt.

Notizen des Teams:

- Pakete installieren (NuGet-Paket Verwalter
-- Core, SqlServer, Designer
- Server erstellen
- Connection String hinterlegen (appsettings.json)

User-Stories

Epic: #2 - Datenbank

Titel: 2.2 - Code-First

Story:

Als Entwickler möchte ich, dass die Web-Anwendung nach den Code-First-Ansatz entwickelt wird, damit ich das Datenmodell schrittweise entwickeln und die Datenbank mit Migrations daran anpassen kann.

Akzeptanzkriterien:

- Es wird eine DbContext-Klasse erstellt.
- Die DbContext-Klasse wird als Service in der Klasse Startup registriert.
- Es gibt eine statische SeedData-Klasse, die dafür sorgt, dass die Migrations ausgeführt werden.

Notizen des Teams:

- Interface für Dependency Injection erstellen
- Entity Framework Klasse erstellen
- DBContext klasse erstellen
- Scope in Program.cs registrieren
- statisches SeedData Model (Dummy) erstellen
- SeedData für Migrations registrieren

User-Stories

Epic: #3 - Admin-Bereich

Titel: 3.1 - Admin-Startseite

Story:

Als Admin möchte ich, dass es einen eigenen Admin-Bereich in der Anwendung gibt, damit ich Stammdaten für die Anwendung bearbeiten kann.

Akzeptanzkriterien:

- Der Admin-Bereich ist über die URL /Admin/Home erreichbar.
- Der Admin-Bereich nutzt eine eigene Layout-Datei.
- Auf der Startseite der Anwendung gibt es keinen Link zum Admin-Bereich, im Admin-Bereich gibt es aber einen Link zur Startseite.

Notizen des Teams:

- Ordnerstruktur für Views anlegen
- AdminController hinzufügen
- _AdminLayout anlegen
- Endpoints in Program.cs anlegen
- Views für Adminbereich anlegen

User-Stories

Epic: #3 - Admin-Bereich

Titel: 3.2 - Projekte CRUD

Story:

Als Admin möchte ich, dass es im Admin-Bereich einen Unterbereich für Projekte gibt, damit ich Projekte ansehen, anlegen, bearbeiten und evtl. löschen kann.

Akzeptanzkriterien:

- Eine Entität `Project` besteht aus den Attributen `Id`, `Title`, `Description`, `Start` und `End`.
- Die `Id` soll automatisch von der Datenbank vergeben und vom Benutzer nicht verändert werden können.
- Das Attribut `End` muss auch `Null`-Werte enthalten können.

Notizen des Teams:

User-Stories

Epic: #3 - Admin-Bereich

Titel: 3.3 - Mitarbeiter CRUD

Story:

Als Admin möchte ich, dass es im Admin-Bereich einen Unterbereich für Mitarbeiter gibt, damit ich Mitarbeiter ansehen, anlegen, bearbeiten und evtl. löschen kann.

Akzeptanzkriterien:

- Eine Entität `Employee` besteht aus den Attributen `Id`, `FirstName`, `LastName` und `JobTitle`.
- Die `Id` soll automatisch von der Datenbank vergeben und vom Benutzer nicht verändert werden können.
- Als `JobTitle` dürfen nur die Werte Developer und Tester eingetragen werden.

Notizen des Teams:

User-Stories

Epic: #4 - Ticket-Bereich

Titel: 4.1 - Ticketliste

Story:

Als Benutzer möchte ich, dass es eine Übersicht aller angelegten Tickets gibt, damit ich auf einen Blick sehen kann, welche Tickets es im System gibt.

Akzeptanzkriterien:

- Eine Entität `Ticket` besteht aus den Attributen `Id`, `Headline`, `Description`, `Project`, `CreatedAt`, `CreatedBy`, `AssignedTo` und `SolvedAt`.
- Die `Id` soll automatisch von der Datenbank vergeben und vom Benutzer nicht verändert werden können.
- Das Attribut `Project` referenziert eine gültige `Project`-Entität.
- Die Attribute `CreatedBy` und `CreatedAt` referenzieren jeweils eine gültige `Employee`-Entität.
- Das Attribut `SolvedAt` muss auch Null-Werte enthalten können.
- Die Tickets werden nur mit den Attributen `Project` und `Headline` sortiert nach `Project` angezeigt.

Notizen des Teams:

User-Stories

Epic: #4 - Ticket-Bereich

Titel: 4.2 - Detailseite

Story:

Als Benutzer möchte ich, dass es eine Detailseite für Tickets gibt, damit ich alle Angaben zu einem einzelnen Ticket einsehen kann.

Akzeptanzkriterien:

- Die Detailseite kann durch Anklicken eines Tickets in der Ticketliste erreicht werden.
- Auf der Detailseite werden alle Angaben zum Ticket übersichtlich dargestellt.
- Es gibt einen Button, mit dem man zur Übersicht der Tickets zurück gelangt.

Notizen des Teams:

User-Stories

Epic: #4 - Ticket-Bereich

Titel: 4.3 - Ticket anlegen

Story:

Als Benutzer möchte ich, dass es ein Formular zur Eingabe eines neuen Tickets gibt, damit ich neue Tickets im System anlegen kann.

Akzeptanzkriterien:

- Das Formular zur Anlage eines neuen Tickets kann über einen Button auf der Ticketliste erreicht werden.
- Die Attribute `Headline`, `Description`, `CreatedBy` und `Project` sind Pflichtfelder.
- Das Attribut `CreatedAt` wird automatisch vom System gesetzt.
- Bei fehlerhaften Eingaben soll der Benutzer mit Fehlermeldungen darauf hingewiesen werden.

Notizen des Teams:

User-Stories

Epic: #4 - Ticket-Bereich

Titel: 4.4 - Ticket bearbeiten

Story:

Als Benutzer möchte ich, dass es ein Formular zur Bearbeitung eines bestehenden Tickets gibt, damit ich Tickets auch später noch verändern kann.

Akzeptanzkriterien:

- Das Formular zur Bearbeitung eines Tickets kann über einen Button neben dem Ticket in der Ticketliste erreicht werden.
- Zur Bearbeitung sollen nur die beiden Attribute `Description` und `AssignedTo` freigegeben sein. Die anderen Attribute sollen nur angezeigt werden.
- Das Attribut `Description` ist ein Pflichtfeld.
- Bei fehlerhaften Eingaben soll der Benutzer mit Fehlermeldungen darauf hingewiesen werden.

Notizen des Teams:

User-Stories

Epic: #4 - Ticket-Bereich

Titel: 4.5 - Ticket lösen

Story:

Als Benutzer möchte ich, dass es eine Möglichkeit gibt ein Ticket als gelöst zu markieren, damit man offene von bereits gelösten Tickets unterscheiden kann.

Akzeptanzkriterien:

- In der Detailansicht eines Tickets soll es einen Button geben, der das Ticket als gelöst markiert.
- Durch Anklicken des Buttons wird automatisch das Attribut `SolvedAt` gesetzt.
- Wenn das Ticket als gelöst markiert ist, kann es nicht mehr bearbeitet werden.

Notizen des Teams:

User-Stories

Epic: #5 - Startseite

Titel: 5.1 - Startseite

Story:

Als Benutzer möchte ich, dass es in der Anwendung eine Startseite gibt, damit ich einen definierten Einstiegspunkt in die Anwendung habe.

Akzeptanzkriterien:

- Die Startseite zeigt den Namen der Anwendung, ein Logo und das aktuelle Datum an.
- Die Startseite wird über die Default-URL (/Home/Index) aufgerufen.
- Zur Ticketliste gelangt man über einen Button.

Notizen des Teams:

User-Stories

Epic: #5 - Startseite

Titel: 5.2 - Statistiken

Story:

Als Benutzer möchte ich, dass auf der Startseite Statistiken über die in der Anwendung enthaltenen Daten angezeigt werden, damit ich mir auf einen Blick ein Bild über den Status der Anwendung machen kann.

Akzeptanzkriterien:

- Es gibt einen Bereich zu den Tickets, wo die Anzahl der Tickets insgesamt und die Anzahl der offenen und geschlossenen Tickets angezeigt wird.
- Es gibt einen Bereich zu den Mitarbeitern, wo die Anzahl der Mitarbeiter insgesamt und die Anzahl der Developer und Tester angezeigt wird.
- Es gibt einen Bereich zu den Projekten, wo die Anzahl der Projekte insgesamt und die Anzahl der abgeschlossenen und noch laufenden Projekte angezeigt wird.

Notizen des Teams:

Ihre Aufgaben

1. Bilden Sie 4er-Gruppen.
   - Sinnvollerweise sollten die Gruppen so gewählt werden, dass jedes Team die Chance hat die Aufgabe umzusetzen.
   - Arbeiten Sie mit einer Code-Basis. Das Programmieren geschieht aber wechselweise und jeder ist mal an der Reihe.

2. Schreiben Sie einen Projektantrag nach IHK-Vorgabe.
   - Eine Vorlage zum Ausfüllen finden Sie in den Projekt-Unterlagen.
   - Das gesamte Team schreibt am Antrag mit.

Ihre Aufgaben

3. Lassen Sie den Projekt-Antrag durch den Dozenten freigeben.
   - Senden Sie den Antrag als PDF per E-Mail an den Dozenten.
   - Sollte der Antrag nicht freigegeben werden, überarbeiten Sie ihn entsprechend der gemachten Auflagen.

4. Starten Sie mit dem Projekt.
   - Vereinbaren Sie mit dem Dozenten einen Zeitpunkt für Ihr tägliches Standup-Meeting.
   - Das Standup-Meeting ist Pflicht für alle Team-Mitglieder.
   - Bearbeiten Sie die User-Stories in der angegebenen Reihenfolge.

Ihre Aufgaben

5. Setzen Sie die erste User-Story um.

- Besprechen Sie im Team, wie die User-Story umgesetzt werden kann. Überlegen Sie, welche Teil-Aufgaben daraus resultieren. Planen Sie, wie Sie die einzelnen Aufgaben umsetzen wollen.
- Vereinbaren Sie einen Termin mit dem Dozenten und besprechen Sie Ihren Plan und klären Sie offene Fragen.
- Setzen Sie die User-Story gemäß Ihrem Plan um.
- Vereinbaren Sie anschließend einen Termin mit dem Dozenten und lassen Sie das Ergebnis Ihrer Arbeit abnehmen.
- Überarbeiten Sie ggf. Ihr Projekt, falls die User-Story nicht abgenommen wurde und präsentieren Sie anschließend erneut.

Ihre Aufgaben

6. Setzen Sie die weiteren User-Stories um.
    - Verfahren Sie mit den weiteren User-Stories wie unter Punkt 5.