Projektarbeit BTS

Ein Bild, das Schrift, Screenshot, Logo, Grafiken enthält.

Automatisch generierte BeschreibungName der Projektarbeit: Ticket System

Abgabedatum: 23.12.2023

Schule: Christian-Schmidt-Schule

Lehrkraft: Heiko Zimmermann

Namen: Vinita Ganeshekumar und David-Ioannis Gugea

Eidesstaatliche Erklärung

Hiermit erklären wir, Vinita Ganeshekumar und David-Ioannis Gugea, dass wir die vorliegende Projektarbeit mit dem Thema „Ticket System“ selbstständig verfasst und nichts als die angegebenen Hilfsmittel benutzt haben. Die Stellen der Arbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angabe der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Grafiken oder bildliche Darstellungen, sowie für Quellen des Internets. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen und wurde auch noch nicht veröffentlicht.

Ort Christian-Schmidt-Schule, Odenwaldstraße 5, 74172 Neckarsulm

Datum 16.12.2023

Unterschriften von David-Ioannis Gugea und Vinita Ganeshekumar

Inhaltsverzeichnis

[1. Zielsetzung des Projektarbeits 4](#_Toc151641811)

[1.1 Pflichtenheft 4](#_Toc151641812)

[1.1.1 Zielbestimmungen 4](#_Toc151641813)

[1.1.2 Einsatz 4](#_Toc151641814)

[1.1.3 Produktübersicht 5](#_Toc151641815)

[1.1.4 Funktionen 5](#_Toc151641816)

[1.1.5 Qualitätsanforderungen 6](#_Toc151641817)

[1.1.6 Benutzeroberfläche 6](#_Toc151641818)

[1.1.7 Technisches Umfeld 6](#_Toc151641819)

[2. Projektplanung 6](#_Toc151641820)

[2.1 Gant-Diagramm, Meilensteine 6](#_Toc151641821)

[2.2 Datenbank ERM 6](#_Toc151641822)

[2.3 Programmiersprachen und Technologien 6](#_Toc151641823)

[3. Praktische Realisierung 7](#_Toc151641824)

[3.1 Containers 7](#_Toc151641825)

[3.2 Datenbank 7](#_Toc151641826)

[3.3 Backend 7](#_Toc151641827)

[3.4 Frontend 7](#_Toc151641828)

[4. Inbetriebnahme 7](#_Toc151641829)

[4.1 Voraussetzungen für die Inbetriebnahme 7](#_Toc151641830)

[4.2 Inbetriebnahme Schritte 7](#_Toc151641831)

[5. Ergebnis 7](#_Toc151641832)

[5.1 Fazit 7](#_Toc151641833)

[5.2 Ausblick 7](#_Toc151641834)

# Zielsetzung des Projektarbeits

## Pflichtenheft

### Zielbestimmungen

Die folgenden Features sind Teil der MUSS-Kriterien:

* Registrierung und Anmeldung von Benutzern
  + Registrierung mit Passwort Bestätigung und Validierung
* Anmeldung mit Benutzername und Passwort
* Benutzerrollen erstellen („admin“, „customer“ und „developer“)
* Tickets erstellen (mit Beschreibung)
* Tickets löschen
* Ticket Beschreibung ändern
* Tickets abschließen
  + Der Benutzer, der das Ticket abgeschlossen hat und das Datum, wann es abgeschlossen wurde, müssen ebenfalls abgespeichert werden
* Eine übersichtliche Tabelle für Tickets
  + Tickets durch Eingabe der Beschreibung in der Suchleiste finden

Abgrenzungskriterien:

* Wenn ein Benutzer sich noch nicht angemeldet hat, darf er keinen Zugriff auf dem Ticket System haben. Zuerst folgt immer eine Anmeldung.
* Passwörter dürfen nicht im Klartext in der Datenbank gespeichert werden.

### Einsatz

Das Ticket-System kann in jedem Software-Entwicklungsprozess eingesetzt werden. Das gesamte Ticket-System kann containerisiert werden, so dass es in jeder Cloud-Umgebung laufen kann, was es sehr flexibel und einfach in der Anwendung macht. Die Datenbankvolumen können auch von Dritten verwaltet werden, was Backups erleichtert und das Datenbanksystem als Ganzes zuverlässiger macht. Das zugrundeliegende Datenbanksystem, PostgreSQL, kann auch mehrere Verbindungen gleichzeitig verarbeiten, so dass mehrere Nutzer gleichzeitig mit der Datenbank verbunden sein können, ohne sich um Race Conditions usw. kümmern zu müssen, da alles vom Server verwaltet wird.

### Produktübersicht

Ohne Anmeldung ist der Zugriff auf das Ticket-System nicht möglich. Die Erstellung von Tickets erfolgt durch Hinzufügen einer Beschreibung. Benutzer der Gruppe haben die Befugnis, Tickets zu schließen und deren Beschreibung kann jederzeit geändert werden. Das Abschlussdatum eines Tickets wird protokolliert und alle Tickets, unabhängig vom Ersteller, sind in einer klaren Tabelle einsehbar.

### Funktionen

In den folgenden Punkten, erfolgt die Erklärung jedes einzelnen Anwendungsfalls.

* Benutzerregistrierung: Wenn eine Person im System noch nicht angemeldet ist, muss ein neues Konto eröffnet werden. Das Frontend wird dann ein Request zu dem Backend senden, um zu überprüfen, ob die Benutzername und das Passwort gültig sind. Falls der eingetragene Benutzername und/oder Passwort sich bereits im Datenbanksystem befinden, wird eine Fehlermeldung angezeigt. Der Benutzer muss ein Konto mit einer neuen Benutzername erstellen. Im Frontend wurde zur Sicherstellung eine Passwort Bestätigung und Validierung erstellt, dass der Benutzer das richtige Passwort eingibt und den Sicherheitsrichtlinien entspricht.
* Benutzeranmeldung: Der Benutzer trägt die Benutzername und das Passwort in den Inputfeldern ein und drückt auf „Login“. Das Backend wird dann überprüfen, ob die Benutzername und das Passwort gültig sind. Wenn sie gültig sind, bekommt der Benutzer Zugriff auf dem Ticket System. Wenn nicht, bekommt der Benutzer eine Fehlermeldung.
* Benutzerrollen erstellen: Bei der Registrierung erfolgt automatisch die Zuweisung des Benutzerrolle „admin“, um eine umfassende Webanwendung mit allen Rechten zu ermöglichen. In der Regel wird jedem neu registrieren Benutzer der Benutzerrolle „customer“ zugewiesen. Nach einloggen als „admin“ können die Benutzerrollen für die Benutzern zugeteilt werden. Dies durch eine Benutzertabelle sichtlich wird.
* Benutzertabelle lesen: Nur der Benutzer, der als „admin“ angemeldet ist, kann die Benutzertabelle sehen und die Rollen verteilen.
* Tickets lesen und bearbeiten: Der „admin“ und „developer“-Benutzern können alle Tickets in der Tabelle lesen und bearbeiten. Der Benutzer „customer“ kann nur seine Tickets lesen und bearbeiten.
* Ticket erstellen: Für die Erstellung eines Tickets, muss eine Beschreibung eingefügt werden. Das Frontend schickt dann zum Backend im Request Body die ID des eingeloggten Benutzers und die Beschreibung des Tickets. Somit lässt sich sicherstellen, wer das Ticket erstellt hat, wann, und worum es im Ticket geht. Das Datum, wann das Ticket erstellt wurde, wird im Backend automatisch gespeichert.
* Ticket Beschreibung ändern: Die neue Beschreibung des Tickets wird, zusammen mit dem Ticket ID, im Request Body abgespeichert und dann zum Backend abgeschickt. Das Backend wird dann die Beschreibung mit Hilfe eines ORMs.
* Ticket abschließen: Wenn ein Ticket abgeschlossen werden muss, wird das Backend die Ticket id, und die Benutzer id von dem Benutzer, der das Ticket abschließen will, brauchen. Das Backend wird dann alle Daten in der Datenbank abspeichern, inklusiv, das Datum, wann ein Ticket abgeschlossen wurde. Alle anderen Benutzer werden dann sehen können, wer das Ticket abgeschlossen hat, und wann es abgeschlossen wurde.
* Ticket löschen: Um ein Ticket komplett löschen zu können, wird das Backend die Ticket ID von dem Frontend benötigen. Diese Funktion kann nicht rückgängig gemacht werden. Der „admin“ und „developer“-Benutzern können alle Tickets in der Tabelle löschen. Der Benutzer „customer“ kann nur seine Tickets löschen.

### Qualitätsanforderungen

|  |  |
| --- | --- |
| Änderbarkeit | Sehr gut |
| Effizienz | Sehr gut |
| Zuverlässigkeit | Sehr gut |
| Sicherheit | Gut |
| Benutzerfreundlichkeit | Sehr gut |
| Skalierbarkeit | Sehr gut |
| Wartbarkeit | Gut |
| Kompatibilität | Nicht relevant |
| Testbarkeit | Gut |
| Datenintegrität | Sehr gut |
| Performanz | Gut |

### Benutzeroberfläche

Die Benutzeroberfläche ist in zwei Hauptteile unterteilt: das Anmelde- und Registrierungsformular sowie das Ticketsystem. Die gesamte Anwendung wurde mithilfe des Vue.js-Frameworks entwickelt, das die Umsetzung von Single-Page-Anwendungen (SPAs) ermöglicht. Die Hauptvorteile von Single-Page-Anwendungen sind die schnelle Benutzererfahrung, die reibungslose Benutzerinteraktion, die reduzierte Serverlast, der verbesserte Arbeitsablauf für Entwickler und die stark verbesserte Benutzereinbindung.

### Technisches Umfeld

Die Softwaresysteme, die gebraucht werden, sind python und docker. Die docker compose Datei im root startet des gebrauchten Containers für das backend. Mit venv kann dann eine virtuelle Umgebung starten und die benötigten python libraries installieren, um den Backendserver starten zu können.

# Projektplanung

## Gant-Diagramm

A green line on a white grid

Description automatically generated

## Datenbank ERM

A graph with a line and a sign

Description automatically generated with medium confidence

## Programmiersprachen und Technologien

Das Projekt verwendet Fast-API als Backend Framework. Fast-API verfügt über API-Dokumentation, d. h. es generiert automatisch OpenAPI- und JSON-Schema-Dokumentation für die API auf der Grundlage der Python-Typ-Hinweise. Außerdem bietet es Typsicherheit, was bei der Erkennung von Fehlern und bei der Nutzung der API sehr hilfreich ist. Die einfache Integration mit pydantic macht es leicht, Daten zu validieren und Datenmodelle mit Validierungsregeln zu definieren und diese für kommende Anfragen zu verwenden.

SQLAlchemy ist das ORM (Object-Relational-Mapping), das im Projekt benutzt wird, da es sich zu einem Industriestandard entwickelt hat und sehr einfach in FastAPI zu integrieren ist. Es macht die Interaktion mit relationalen Datenbanken sehr einfach und flexibel und ist besonders nützlich, wenn es um die Abstraktion von Datenbanken geht, da es mehrere Datenbank-Backends unterstützt und eine konsistente API unabhängig vom zugrunde liegenden Datenbanksystem bietet.

PostgreSQL wird als Datenbank benutzt, weil es Open Source ist und von der Community sehr gut unterstützt wird. Außerdem ist es sehr skalierbar und verfügt über eine Gleichzeitigkeitskontrolle für den Fall, dass mehrere Clients an die Datenbank angeschlossen sind. Die Postgresql-Datenbank und den pg-Admin-Browser wird in Docker-Containern ausgeführt. Ein Docker-Container ist ein leichtgewichtiges, portables Paket, das alles enthält, was zum Ausführen einer Software benötigt wird. Container sind wie virtuelle Maschinen, aber effizienter, da sie den Kernel des Host-Betriebssystems gemeinsam nutzen und gleichzeitig eine Isolierung für die Anwendung bieten. Das wird aus Gründen der Isolation und Portabilität getan. Durch die Verwendung der Docker-Engine ist es einfach und flexibel, sie überall zu starten, was die Entwicklung einfacher und schneller macht.

Vue.js wird als Frontend-Framework verwendet, um eine dynamische und reaktive Benutzeroberfläche zu gestalten. Es ermöglicht die einfache Integration von Komponenten, Datenbindung und das effiziente Handling von Benutzerinteraktionen, was zu einer ansprechenden und performanten Webanwendung führt.

# Praktische Realisierung

## Containers

Ein Container ist ein Linux-Prozess, der durch cgroups und Namespaces verwaltet wird. Docker verwendet ein Linux-Subsystem auf Windows, um die Verwendung von Containern auf Windows-Systemen zu ermöglichen.

Ein Prozess ist die Instanz eines Programms, das gerade ausgeführt wird. Jeder Prozess enthält seinen eigenen Heap (Speicher). Ein Prozess enthält auch Prozessregister, Programmzähler, Stapelzeiger und, wie bereits gesagt, "Speicherseiten" (d.h. Stapelhaufen, statischer Speicher usw.). Da jeder Prozess seinen eigenen Speicheradressraum (Heap) hat, bedeutet dies auch, dass ein Prozess andere Prozesse nicht beschädigen kann, d.h. wenn ein Prozess ausfällt, beschädigt dies nicht andere laufende Prozesse. Ein Prozess kann mehrere andere Prozesse starten, wobei der erstellende Prozess als Elternprozess bezeichnet wird und die von diesem Prozess gestarteten Prozesse als seine Kindprozesse. Jeder erstellte Prozess kann dann wiederum neue andere Prozesse erstellen, die dann einen Baum von Prozessen bilden. Jedes Mal, wenn einen Linux-Rechner gestartet wird, wird ein erster Prozess namens "INIT" gestartet, der die PID (Prozess-ID) 1 erhält. Hier beginnt der Prozessbaum.

Mit Cgroups können Prozesse in Gruppen organisiert werden. Diese Gruppen können dann überwacht werden und es können bestimmte Grenzen für diese Gruppen festgelegt werden (z. B. CPU-/Speichergrenzen). Cgroups werden nicht automatisch erstellt. Eine Cgroup kann für bestimmte Prozesse und den Prozessbaum, der aus ihnen hervorgeht, erstellt werden.

Namensräume werden verwendet, um Prozesse vom Rest zu isolieren. Sie können verwendet werden, um Ressourcen wie Prozess-IDs, Netzwerkschnittstellen usw. zu partitionieren, so dass verschiedene Prozesse ihre eigenen Instanzen dieser Ressourcen haben können, als ob sie in ihrer eigenen unabhängigen Umgebung laufen würden.

Containers, in diesem Projekt, zu benutzen, hat die folgenden Vorteile gehabt:

* Isolierung: Anwendungen können unabhängig voneinander arbeiten, ohne sich gegenseitig zu stören, da das Dateisystem und die Prozesse in Containern voneinander getrennt sind. Dadurch, dass sichergestellt wird, dass Konfigurationen und Abhängigkeiten innerhalb des Containers enthalten sind, trägt diese Isolierung zur Minimierung von Konflikten und Kompatibilitätsproblemen bei.
* Portabilität: Die Anwendung und ihre Abhängigkeiten sind in Containern gekapselt, was den Transport und den Betrieb der Anwendung in verschiedenen Kontexten, einschließlich Entwicklung, Test und Produktion, erleichtert. Das Problem "es funktioniert auf meinem Rechner" wird durch diese Portabilität gelöst, die auch das Bereitstellungsverfahren vereinfacht.
* Schnelles Testen und Feedback: Mit Containern können Entwickler Änderungen lokal in einer produktionsähnlichen Umgebung testen. So können Ideen schnell getestet werden und Erhalt von sofortigem Feedback, was einen iterativen Entwicklungsprozess fördert.

## Datenbank

In einer relationalen Datenbank können Daten aufgrund der Beziehungen zwischen den Daten dank ihres strukturellen Aufbaus angeordnet und zugänglich gemacht werden.

In relationalen Datenbanken werden die Daten in Tabellen gespeichert, die in Zeilen und Spalten angeordnet sind. In einer Tabelle steht jede Zeile für einen Datensatz und jede Spalte für ein Feld oder Attribut des Datensatzes. Gemeinsame Felder (oder Schlüssel) in vielen Tabellen dienen als Grundlage für Beziehungen zwischen ihnen. Dank dieser Verknüpfungen kann die Datenbank verwandte Daten aus anderen Tabellen effektiv miteinander verbinden. Eine Spalte (oder Gruppe von Feldern) in einer Tabelle, die eine eindeutige Identität für jeden Datensatz in der Tabelle bietet, wird als Schlüssel bezeichnet. Es gibt verschiedene Arten von Schlüsseln, z. B. Fremdschlüssel, die zur Erstellung tabellenübergreifender Verknüpfungen verwendet werden, und Primärschlüssel, die zur eindeutigen Identifizierung von Datensätzen innerhalb einer Tabelle dienen.

Die Verwendung einer relationalen Datenbank bietet verschiedene Vorteile, darunter:

* Datenintegrität: Einschränkungen wie Primärschlüssel, Fremdschlüssel und eindeutige Einschränkungen werden von relationalen Datenbanken verwendet, um die Datenintegrität zu gewährleisten. Dies trägt dazu bei, die Daten konsistent und genau zu halten.
* Beziehungen: Relationale Datenbanken erleichtern die Erstellung von Verbindungen zwischen Tabellen. Dies ermöglicht es, komplizierte Datenstrukturen auszudrücken und zugehörige Daten effektiv abzufragen.
* Skalierbarkeit: Relationale Datenbanken können horizontal (durch Verteilen der Datenbank auf mehrere Server) oder vertikal (durch Hinzufügen weiterer Ressourcen zu einem einzigen Server) erweitert werden. Daher können sie steigende Benutzerzahlen und Datenmengen bewältigen.
* ACID-Eigenschaften: Relationale Datenbanken folgen den ACID-Eigenschaften (Atomicity, Consistency, Isolation, Durability), die eine konsistente und verlässliche Transaktionsverarbeitung auch bei Systemausfällen garantieren.

Die Datenbank besteht aus zwei Tabellen: Benutzer und Tickets. Die Tabelle "Benutzer" enthält einen Primärschlüssel, nämlich die ID des Benutzers. Sie enthält außerdem den Benutzernamen, das Passwort, den Benutzerrolle und den Salt-Wert, der zum Hashing des Passworts verwendet wird. Beim Hashing werden die Eingabedaten, die häufig eine variable Größe haben, in eine Zeichenkette fester Länge umgewandelt, die in der Regel als zufällige Folge von Buchstaben und Ziffern erscheint. Eine Methode zur Erhöhung der Sicherheit von gehashten Kennwörtern ist das Salting. Eine zufällige Datenfolge, die speziell für jeden Benutzer erstellt wird, wird als Salt bezeichnet. Das Kennwort des Benutzers und das Salt werden dann vor dem Hashing gemischt. Das Salt und der generierte Hash werden in der Datenbank gespeichert.

Die Ticket-Tabelle enthält die ID des Tickets, die ein Primärschlüssel ist. Sie enthält die Beschreibung, das Erstellungs- und Abschlussdatum sowie einen booleschen Wert namens "done", der anzeigt, ob das Ticket abgeschlossen wurde oder nicht. Sie enthält auch zwei Primärschlüssel zur Benutzertabelle, "closed\_by", der auf die ID des Benutzers verweist, der das Ticket geschlossen hat, und "Benutzer\_id", der auf den Benutzer verweist, der das Ticket erstellt hat.

## Backend

Mit Hilfe des Python Fast-Api-Frameworks bietet das Backend eine Restful-API. Die Representational State Transfer Application Programming Interface (REST API) ist ein architektonischer Ansatz für die Entwicklung von Netzwerkanwendungen. Es handelt sich um eine Sammlung von Richtlinien und Beschränkungen, die bei der Entwicklung von Online-Diensten verwendet werden. REST ist eine Reihe von Prinzipien, die angeben, wie Webdienste erstellt werden sollten; es ist weder ein Standard noch ein Protokoll an sich.

Zu den wichtigsten Grundsätzen und Merkmalen einer REST-API gehören:

* Zustandslosigkeit: Jede Anfrage, die ein Client an einen Server sendet, muss alle Details enthalten, die der Server benötigt, um die Anfrage zu verstehen und zu beantworten. Zwischen den Anfragen sollte der Server den Status des Clients nicht verfolgen.
* Client-Server-Architektur: Durch die Verwendung eines Netzwerks zur Erleichterung der Kommunikation sind der Client und der Server unabhängige Einheiten. Während der Server für die Bearbeitung von Anfragen und die Ressourcenverwaltung zuständig ist, ist der Client für die Benutzeroberfläche und die Benutzererfahrung verantwortlich.
* Einheitliche Schnittstelle: Eine konsistente und einheitliche Schnittstelle ist ein Muss für REST-APIs. Dies umfasst die Verwendung gängiger HTTP-Methoden (GET, POST, PUT, DELETE) sowie eine Reihe von Regeln für die Darstellung und Benennung von Ressourcen.

Ein Web-Framework ist eine Softwarearchitektur, die die Entwicklung von Webanwendungen unterstützen soll. Dazu bietet es vorgefertigte Teile und Dienstprogramme zur Vereinfachung von Routinevorgängen wie der Beantwortung von HTTP-Anfragen, der Pflege von Datenbanken und der Vereinfachung der Gestaltung von Benutzeroberflächen. Web-Frameworks bieten eine definierte Basis und wiederverwendeten Code, damit Entwickler Webanwendungen schneller erstellen können.

Die Hauptgründe, warum fast api benutzt wird, sind die automatische Dokumentation, die Handhabung von Typ-Annotationen und die Erleichterung der Datenvalidierung und Serialisierung mit Pydantic, was den Entwicklungsprozess rationalisiert. Außerdem ermöglicht sie Dependency Injection, was die saubere, modulare Organisation und Verwaltung von Abhängigkeiten vereinfacht.

Ein ORM wurde verwendet und in das Web-Framework integriert, um mit der Postgresql-Datenbank zu arbeiten. ORM steht für Object-Relational Mapping. Es handelt sich um ein objektorientiertes Programmierparadigma und eine Programmiertechnik, die die Lücke zwischen relationalen Datenbanken und Software-Entwurfsmustern in der Softwareentwicklung schließt.

## 3.4 Frontend

Das Frontend des Projekts basiert auf dem vielseitigen Vue.js-Framework, einem JavaScript-Framework für zeitgemäße Benutzeroberflächen. Vue.js zeichnet sich durch klare Struktur, einfache Integration und effiziente Datenbindung aus, was die Erstellung interaktiver Webanwendungen erleichtert. Das Framework ermöglicht die Entwicklung nahtloser Benutzererfahrungen, insbesondere durch das schnelle Laden von Seiteninhalten ohne vollständiges Neuladen der Seite, was die Benutzerfreundlichkeit verbessert, und die Serverlast reduziert.

Der komponentenbasierte Ansatz von Vue.js vereinfacht die Entwicklung und Wartung von Code, da wiederverwendbare Komponenten effizient in verschiedenen Teilen der Anwendung integriert werden können. Dies fördert einen reibungslosen Entwicklungsprozess und gewährleistet eine konsistente Struktur im Frontend-Code. Die reaktive Datenbindung von Vue.js ermöglicht die nahtlose Übertragung und Aktualisierung von Daten zwischen Komponenten, erleichtert die Synchronisation von Frontend- und Backend-Daten.

Zusätzlich zu Vue.js werden HTML und CSS für die Gestaltung der Benutzeroberfläche verwendet, was zu einem ansprechenden und benutzerfreundlichen Design beiträgt. Insgesamt bietet die Kombination aus Vue.js, HTML und CSS eine solide Grundlage für die Entwicklung eines zeitgemäßen, den aktuellen Standards entsprechenden Frontends mit Fokus auf Flexibilität, Performance und Entwicklerfreundlichkeit.

Die wichtigsten Hauptkomponenten aus dem Frontend sind:

* public: Dieses Verzeichnis enthält statische Ressourcen wie Bilder, die direkt über das Web aufgerufen werden können. In diesem Projekt enthält es den Index.html-Datei, die als Hauptpunkt für den Zugriff auf die Anwendung fungiert.
* assets: Dieser Ordner enthält Bilder, Schriftarten und andere Ressourcen, die von der Anwendung verwendet werden.
* router: Die "router"-Dateien enthalten die Logik zur Navigation zwischen verschiedenen Ansichten in der Anwendung.
* views: Die "views"-Dateien enthalten die Vue-Komponenten, die den Inhalt der einzelnen Ansichten darstellen.
* App.vue: Die App.vue-Datei ist die Haupt-Vue-Komponente, die die gesamte Anwendung darstellt.
* main.js: Die main.js-Datei ist der Einstiegspunkt der Anwendung. Eine neue Vue-Instanz wird erstellt und das HTML-Dokument eingefügt.
* vue.config.js: Diese Datei enthält Konfigurationseinstellungen für die Vue-CLI-Anwendung. Sie kann zum Beispiel verwendet werden, um den Entwicklungsserver, die Build-Optionen oder die Runtime-Konfiguration anzupassen.
* node\_modules: Dieses Verzeichnis enthält alle von yarn installierten Pakete und Abhängigkeiten. Es sollte nicht direkt bearbeitet werden, da es automatisch durch yarn generiert wird.

Der Frontend besteht aus zwei Benutzeroberflächen, das wäre der „HomeView“ und „TicketDashboardView“. Der „HomeView“ beeinhaltet eine ansprechende Anmeldeseite mit verschiedenen Funktionen. Die Seite bietet zwei Ansichten: "Login" und "Register", die durch Klicken auf entsprechende Schaltflächen umgeschaltet werden können. Das Anmeldeformular enthält Benutzername und Passwort, wobei die Passwortstärke überprüft wird. Fehlermeldungen werden angezeigt, wenn erforderliche Felder leer sind oder wenn es Probleme bei der Anmeldung oder Registrierung gibt. Ein Canvas-Element wird für animierte Partikeleffekte verwendet, die auf die Mausbewegung reagieren. Die Partikel bewegen sich entsprechend der Mausbewegung und erzeugen ein ansprechendes visuelles Feedback. Axios wird für HTTP-Anfragen an einen Server verwendet, um Anmeldedaten zu überprüfen und Benutzer zu registrieren. Die Kommunikation mit dem Server erfolgt über RESTful-API-Endpunkte. Das Styling der Seite ist ansprechend gestaltet und umfasst verschiedene Animationen, Farbübergänge und Schatten, um eine ästhetisch ansprechende Benutzeroberfläche zu schaffen. Zusammengefasst beinhaltet der „HomeView“ eine interaktive Registrier- und Anmeldeseite mit visuellen Effekten, einer klaren Benutzerführung und Validierungen für eine verbesserte Benutzererfahrung.

Der „TicketDashboardView“ befasst ein Dashboard mit Funktionen zur Anzeige, Erstellung, Bearbeitung und Löschung von Tickets. Es beinhaltet eine Navigation mit einem Dropdown-Menü für Benutzeraktionen wie Logout und Benutzerverwaltung. Außerdem gibt es eine Suchleiste zur Filterung der Ticketliste.

Die Hauptfunktionalitäten umfassen das Anzeigen von Ticketinformationen wie ID, Beschreibung, Erstellungs- und Abschlussdatum, Ersteller und Status. Benutzer können Tickets bearbeiten, schließen oder löschen. Es gibt auch Pop-up-Formulare für die Ticketerstellung und die Verwaltung von Benutzerdaten, einschließlich der Aktualisierung von Benutzerrollen.

Die Daten werden über Axios-Anfragen an eine lokale API gesendet und empfangen. Zudem wird die lokale Speicherung für Tickets genutzt. Die Oberfläche ist responsiv gestaltet und bietet Funktionen basierend auf dem eingeloggten Benutzerrolle, wie z.B. das Ausblenden von Optionen für bestimmte Benutzerrolle wie „customer“.

Ebenfalls enthält die Seite auch ein Spinner und zeigt Ladevorgänge an. Das Design ist hauptsächlich dunkel gehalten, mit klaren Karten für die Darstellung von Tickets und Benutzerinformationen.

# Inbetriebnahme

## Voraussetzungen für die Inbetriebnahme

Die Software, die zum Starten des Backends benötigt wird, ist Docker Desktop und Python.

Für den Frontend wird das Yarn Packetmanager benötigt, um den Framework Vue.js zu instrallieren.

## .Inbetriebnahme Schritte

Um das Backend zu starten, müssen der Docker-Desktop und Python installiert sein. Anschließend muss im Stammverzeichnis der Anwendung der Befehl "docker-compose up -d" ausgeführt werden, um die für die Datenbank benötigten Container zu starten. Nach dem Start des Containers muss das venv in dem Ordner aktiviert werden, in dem das Backend gespeichert wurde. Nach der Aktivierung von venv müssen alle Pakete aus der Datei requirements.txt installiert werden. Nach der Installation kann der Backend-Api-Server mit dem Befehl: uvicorn main:app gestartet werden.

Um das Frontend zu starten, sollte diese Schritte befolgt werden:

1. Die Kommandozeile oder das Terminal öffnen
2. Zum Verzeichnis des Frontend-Projekts navigieren
3. Den Befehl „yarn run serve“ ausführen

Dadurch wird der Entwicklungsserver gestartet, und die Anwendung sollte unter „http://localhost:8080/“ verfügbar sein. Davor sollten wie laut der „README.md“, die notwendigen Abhängigkeiten mit dem Befehl „yarn install“ installiert werden.

# Ergebnis

## Fazit

Die Entwicklung und Realisierung des Ticket-Systems basiert auf klaren Zielsetzungen, einer strukturierten Projektplanung und dem Einsatz moderner Technologien. Die Verwendung von Docker-Containern ermöglicht eine flexible und isolierte Bereitstellung, während das Fast-API-Framework, SQLAlchemy-ORM und PostgreSQL-Datenbank eine effiziente Backend-Implementierung gewährleistet. Die Benutzeroberfläche, basierend auf Vue.js, bietet eine ansprechende und benutzerfreundliche Erfahrung. Die Anwendung erfüllt die definierten Zielkriterien, darunter Benutzerverwaltung, Ticketerstellung, -bearbeitung und -löschung. Die Nutzung von RESTful-API-Prinzipien und das ORM erleichtern die Dateninteraktion zwischen Frontend und Backend. Die Verwendung von Containern trägt zur Portabilität und Skalierbarkeit bei.

## Ausblick

Für zukünftige Entwicklungen könnte die Implementierung von zusätzlichen Funktionen und Erweiterungen in Betracht gezogen werden. Dazu gehören möglicherweise Benachrichtigungen, erweiterte Filteroptionen für Tickets, Analyse-Tools für die Ticket-Performance und die Integration von Authentifizierungsmechanismen von Drittanbietern.

Die Sicherheit der Anwendung könnte weiter gestärkt werden, beispielsweise durch die Implementierung von HTTPS, zusätzlichen Sicherheitsprüfungen und Authentifizierungsmethoden. Eine umfassende Testabdeckung sowie regelmäßige Sicherheitsaudits könnten die Stabilität und Integrität der Anwendung verbessern.

Die Skalierbarkeit könnte durch die Implementierung von Lastenausgleich und vertikaler Skalierung optimiert werden. Dies könnte notwendig werden, wenn die Benutzerzahl und Ticketdatenbank wachsen.

Insgesamt bietet die entwickelte Anwendung eine solide Grundlage für ein effizientes und flexibles Ticket-Management-System und bietet Raum für zukünftige Erweiterungen und Verbesserungen.