# Helpdesk, ein Ticketierungssystem

#### Projekt der Oberstufe

Informationstechnischen Assistent(inn)en am

TBK-Solingen

Gruppenmitglieder

[Timon Wegner](mailto:taw.kontakt@gmail.com), [Lia Kaup](mailto:lia@xomo.de), [Jadon Sill](mailto:jadonnoah06@gmail.com), [Timo Szadkowski](mailto:timosz0103@gmail.com), [Maurice Severin](mailto:schueler.mseverin@tbk-solingen.de), [Leon Wilde](mailto:l.wilde@mail.de)

[Helpdesk, ein Ticketierungssystem 1](#_Toc17)

[Projekt der Oberstufe 1](#_Toc18)

[Abkürzungsverzeichnis 3](#_Toc19)

[1. Anforderungskatalog 4](#_Toc20)

[2. Entity Relationship Diagramm 6](#_Toc21)

[3. Relationales Datenbankmodell 7](#_Toc22)

[4. Physischer Datenbankentwurf 8](#_Toc23)

[Verwendung der Webseite 9](#_Toc24)

[Erste Anmeldung 9](#_Toc25)

[Team-Aufgaben Verteilung 10](#_Toc26)

[Team Lead 10](#_Toc27)

[Frontend 10](#_Toc28)

[Backend 12](#_Toc29)

[API Dokumentation 14](#_Toc30)

[Wie spricht man die API an? 14](#_Toc31)

[Antwortstruktur 14](#_Toc32)

[Wo findet man Hilfe? 14](#_Toc33)

[Verwendeter Technologie Stack 15](#_Toc34)

## Abkürzungsverzeichnis

API - Programmierschnittstelle

### 1. Anforderungskatalog

Um den internen Bedarf und die Dokumentation sowie Verwaltung von Tickets zu verbessern, wird ein Ticketierungssystem entwickelt. In erster Linie soll den Mitarbeitern der Firma Easitix ermöglicht werdet Tickets effizient zu erstellen, zu bearbeiten und abzurechnen. Wärend einer vorläufigen Diskussionrunde wurden die grundlegenden Strukturen der Datenbank ausgearbeitet (siehe begleitendes [Diagramm](#_2._Entity_)).  
  
Weitere Besprechnungen ergaben nach Entwicklung eines rudimentären Login Systems das während dieses ersten Prototypen auch die Möglichkeit beigelegt werden kann, dass Kunden sowie deren Mitarbeiter sich anmelden können, um ohne Vermittelung der Firma Easitix Tickets eingereicht werden können.  
  
So sollte der erste Prototyp dieses Systems folgende Funktionen umfassen:

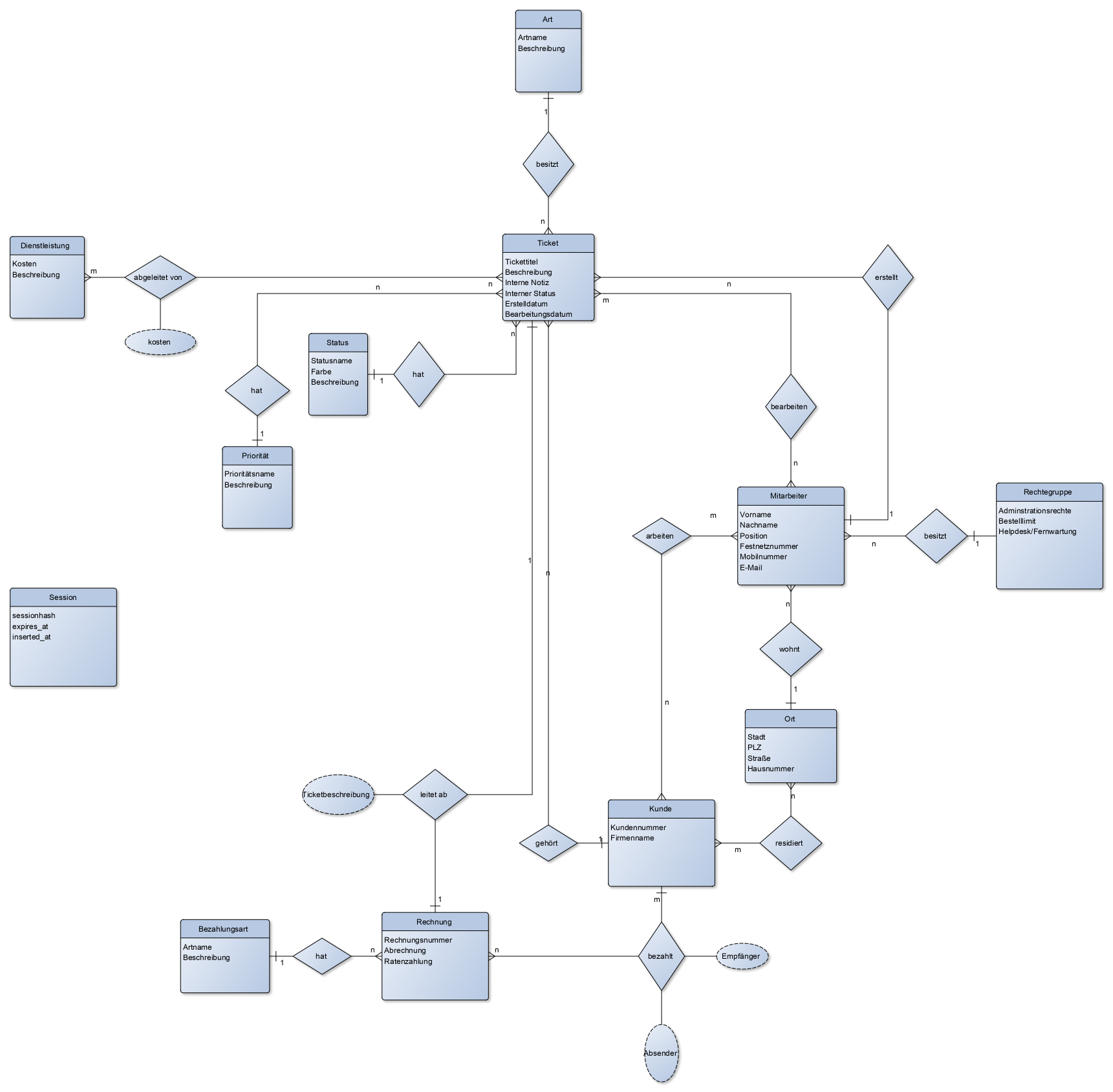
|  |  |  |
| --- | --- | --- |
| **Aufgabe** | **Beschreibung** | **Priorität** |
| **Ticket-Erstellung ermöglichen** | Benutzer müssen neue Tickets für verschiedene Anfragen oder Probleme erstellen können. | Hoch |
| **Statusverwaltung von Tickets** | Tickets sollten unterschiedliche Status (z. B. offen, in Bearbeitung, geschlossen) haben. | Hoch |
| **Detaillierte Beschreibung erfassen** | Jedes Ticket sollte eine detaillierte Problembeschreibung enthalten. | Hoch |
| **Interne Notizen hinzufügen** | System sollte ermöglichen, dass Mitarbeiter interne Notizen zu Tickets hinzufügen können. | Mittel |
| **Prioritäten festlegen** | Tickets müssen priorisiert werden können (z. B. hoch, mittel, niedrig). | Hoch |
| **Zuweisung an Mitarbeiter** | Tickets sollten an bestimmte Mitarbeiter zur Bearbeitung zugewiesen werden können. | Hoch |
| **Kundeninformationen erfassen** | Das System sollte Kundendaten wie Name, E-Mail, Telefonnummer und Kundennummer speichern. | Hoch |
| **Kommunikation mit Kunden** | Das System sollte die Kommunikation zwischen Mitarbeitern und Kunden über das Ticket ermöglichen. | Hoch |

|  |  |  |
| --- | --- | --- |
| **Zeiterfassung und -berechnung** | Es sollte eine Funktion zur Erfassung der Bearbeitungszeit für jedes Ticket geben. | Mittel |
| **Standort- und Firmeninformationen** | Kundenstandorte und Firmenzugehörigkeiten sollten gespeichert werden können. | Niedrig |
| **Ticketabrechnung ermöglichen** | Das System sollte eine Abrechnung der erbrachten Leistungen pro Ticket unterstützen. | Mittel |
| **Rechnungsstellung integrieren** | Rechnungen für bearbeitete Tickets sollten erstellt und versendet werden können. | Mittel |
| **Ratenzahlung verwalten** | Möglichkeit zur Verwaltung von Ratenzahlungen für Rechnungen. | Niedrig |
| **Erstellung von Tickets durch Kunden** | Möglichkeit Tickets zu erstellen ohne einen Mitarbeiter anzufragen diesen zu erstellen | Niedrig |

Zusätzlich wurden nach Rückmeldungen der Kunden folgende weitere Anforderungen definiert:

|  |  |  |
| --- | --- | --- |
| **Aufgabe** | **Beschreibung** | **Priorität** |
| **Erstellung von Berichten** | Das System sollte Berichte über Tickets, deren Status und Bearbeitungszeiten generieren können. | Mittel |
| **E-Mail-Benachrichtigungen** | Benutzer und Kunden sollten E-Mail-Benachrichtigungen bei Ticket-Aktivitäten erhalten können. | Hoch |
| **Such- und Filterfunktionen** | Tickets sollten nach verschiedenen Kriterien wie Status, Priorität und Mitarbeiter gefiltert und durchsucht werden können. | Mittel |
| **Rollen- und Rechteverwaltung** | Es sollten unterschiedliche Benutzerrollen und Zugriffsrechte im System verwaltet werden können. | Hoch |

### 2. Entity Relationship Diagramm



### 3. Relationales Datenbankmodell

abgeleitet: *FK\_Dienstleistung, FK\_Ticket*

arbeitet: *FK\_Mitarbeiter, FK\_Kunde*

art: PK\_Art - Artname, Beschreibung

bearbeitet: *FK\_Mitarbeiter, FK\_Ticket*

bezahlungsart: PK\_Bezahlungsart - Artname, Beschreibung

dienstleistung: PK\_Dienstleistung - Beschreibung, Kosten

kunde: PK\_Kunde - Kundennummer, Firmenname, Email, Passwort

rechnung: Rechnungsnr - Ticketbeschreibung, Abrechnung, Ratenzahlung - *FK\_Bezahlungsart, FK\_Kunde, FK\_Ticket*

mitarbeiter: PK\_Mitarbeiter - Vorname, Nachname, Position, Festnetznummer, Mobilnummer, Email, Passwort - *FK\_Ort, FK\_Rechtgruppe*

ort: PK\_Ort - Stadt, PLZ, Straße, Hausnummer

priorität: PK\_Priorität - Prioritätsname, Beschreibung

rechtegruppe: PK\_Rechtgruppe, - Administrationsrechte, Bestelllimit, Helpdesk\_Fernwartung

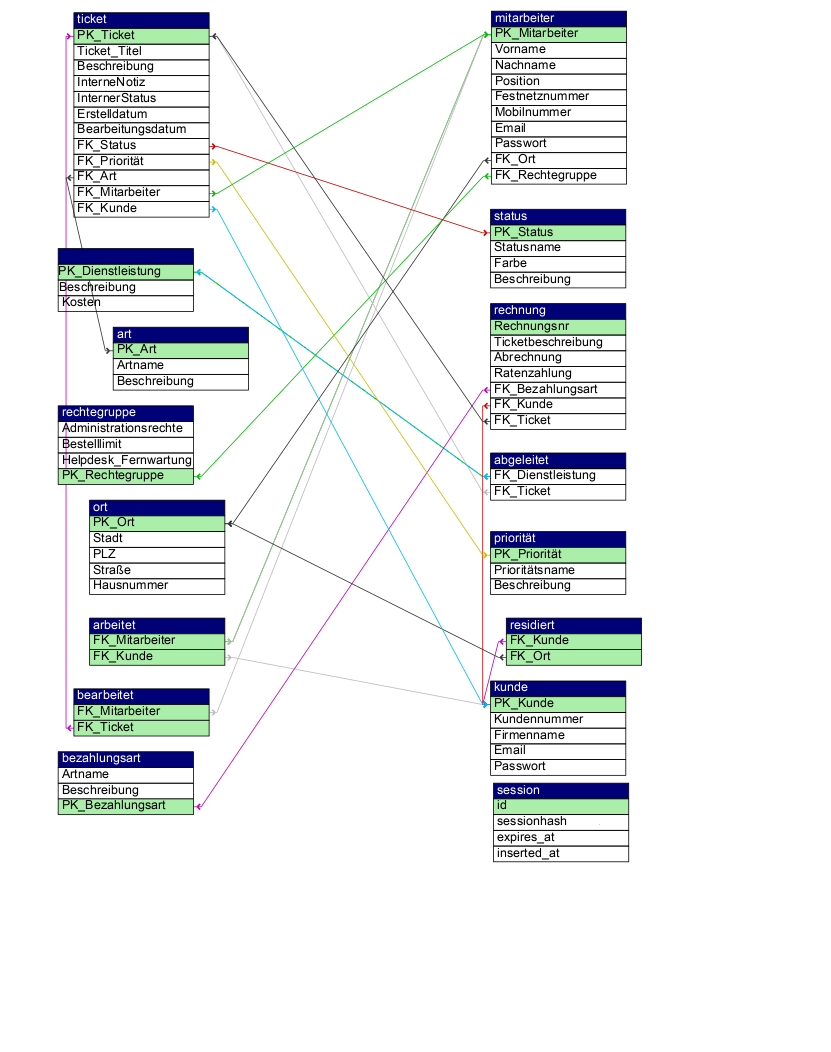
residiert: *FK\_Kunde, FK\_Ort*

session: id - sessionhash, expires\_at, inserted\_at

status: PK\_Status - Statusname, Farbe, Beschreibung

ticket: PK\_Ticket - Ticket\_Titel, Beschreibung, InterneNotiz, InternerStatus, Erstelldatum, Bearbeitungsdatum - *FK\_Status, FK\_Priorität, FK\_Art, FK\_Mitarbeiter, FK\_Kunde*

### 4. Physischer Datenbankentwurf



### Verwendung der Webseite

#### Erste Anmeldung

Gehen Sie auf die von uns gegeben Webseite, melden Sie sich mit den ersten Login Daten aus der E-Mail an und klicken sie in der Navbar auf Login.

Darauffolgendent geben Sie die von uns gegeben Administrationsdaten daraufhin sind Sie in der Hauptseite des Adminpannels und können von hieraus in der Navbar zu allen verschiedenen Unterpunkten, um die für Sie wichtigen Unterpunkte einzutragen. Hierbei sind bereits erste Demo Daten eingetragen.

Sollten hierauffolgended noch weitere Fragen besitzen so Informieren sich uns doch bitte über folgende [EMail](mailto:schueler.twegner@tbk-solingen.de).

### Team-Aufgaben Verteilung

#### Team Lead

Im Team Lead war es die Aufgabe von Timon Andreas Wegner, die Dokumentation zu halten, diese in dieses Dokument zusammen zufassen sowie das vergeben der Aufgaben und nachhalten, ob diese Aufgaben berarbeitet werden und ggf[[1]](#footnote-2). das helfen der Team Mitglieder sofern diese bei einer Aufgabe blockiert sind oder sie nicht bis zur „floating deadline[[2]](#footnote-3)“ bearbeitet haben.

* Aufgaben die Timon Andreas Wegner übernehmen musste:
  + Das Erarbeiten von dem [API](#API) Router
  + Das Überarbeiten der Administrationsüberfläche, da diese nur Serverseitig geschreiben wurde
  + Das Erstellen einiger Endpunkte für die [API](#API)

#### Frontend

Im Frontend waren folgende Team Mitglieder aufgeteilt, Lia Kaup, Timo Szadkowski, Jadon Sill. Die Aufteilung der Aufgaben waren sind so einzuordnen:

* **Lia Kaup**:
  + **Aufgaben**: Erstellen der Homepage, Startpage fürs Adminpannel, Subpages für Datenschutzerklärung, Impressum, ein Navbar Template und ein Footer Template
    - Zusätzlich solle Sie eine Rechnungsübersicht Page vorbereiten für Timo Szadkowski
  + **Bearbeitung dieser Aufgaben (Bewertung durch Team Lead)**:
    - Hat während den ersten Arbeitsphasen relativ gut sich mit dem Thema befasst und sich versucht einzubringen, jedoch war Sie öftermals am Handy und hat sich massivst ablenken lassen durch ein anders Team Mitglied names Leon Wilde. Erwähnenswert ist, dass Sie bevor sie auf den Team Leader bei Problemen zukam diese erstmals versucht hat zu verstehen. Außerdem hat sie für Ihre Aufgaben überdurchschnittlich viel Zeit benötigt.
* **Timo Szadkowski**:
  + **Aufgaben**: Erstellen der Rechnungsbearbeitung, Rechnungserstellung Pages, der Erstellung von PDF‘s aus Rechnungen und Übernahme und Implementierung der Rechnungsübersicht Page von Lia Kaup, zusätzlich auch die Funktionalität aller Pages.
    - Zusätzlich am Anfang hat er sich extremst für die Ausarbeitung der Anforderungen eingesetzt sowie bei der Erstellung des ER Modells, als auch bei Erstellung und Überprüfung des Relationalem Datenbankmodells.
  + **Bearbeitung dieser Aufgaben** **(Bewertung durch Team Lead)**:
    - Hat seine Aufgaben souverän bearbeitet und auch trotz ersten Streitpunkten mit dem Team Leader die Aufgaben nachfolgend weiterhin bearbeitet und konstruktiv bestimmte Entscheidungen infrage gestellt. Bei Problemen auf die er gestoßen ist hat er versucht diese eigenständig zu lösen, mithilfe Hilfsmitteln, wie [Mozilla Development Site](https://developer.mozilla.org/en-US/), [W3-Schools](https://www.w3schools.com/) und [ChatGPT](https://chatgpt.com/). Sollte er, dann diese nicht bearbeiten können ist er konstruktiv mit den bereits erforschten Teilen eigenständig auf den Team Leader zugekommen
* **Jadon Sill:**
  + **Aufgaben:** Erstellung der Ticketerstellung und Bearbeitungs Pages, der Ticketübersicht Page, die Funktionen zur Übergebung des Tickets zur Rechnung, als auch die Funktionalität aller Pages.
    - Zusätzlich am Anfang war er mit Timo Szadkowski und Timon Andreas Wegner ein treibender Fluss um die Struktur der Datenbank festzulegen und hat mit Timo Szadkowski das Relationale Datenbankmodell erstellt.
  + **Bearbeitung dieser Aufgaben** **(Bewertung durch Team Lead)**:
    - Hat seine Aufgaben eigenständig bearbeitet ohne, dass es zu größeren Problemen kam. Zusätzlich hat er sich auch eigenständig mit anderen Gruppenmitglieder verständigt mit Auskunft an den Team Leader über Entscheidungen, damit er diese dokumentiert. Auch bei der Problem Lösung hat er auf die gleichen Seiten, wie bei Timo Szadkowski gelegt und hat auch hier konstruktiv mit dem Team Leader gearbeitet falls, diese Ihm nicht bei der Lösung dieser Probleme geholfen haben.

#### Backend

Das Backend Team bestand aus zwei Mitgliedern offiziell namens Maurice Severin und Leon Wilde. Die Aufgabenverteilung war wie folgt:

* **Leon Wilde:**
  + **Aufgaben:** Erstellung der Datenbank aus ER- und Relationalem Modell, Erarbeitung der gestellten Aufgaben für die API durch den Team Leader, dem Helfen mit der Dokumentation von der API, das Erstellen eines „Admin Panels“[[3]](#footnote-4), sowie das erste Einfügen von Beispiel Daten.
  + **Bearbeitung dieser Aufgaben** **(Bewertung durch Team Lead)**:
    - Während der Bearbeitung der Aufgaben gelang es nicht Leon Wilde konzentriert für eine Zeitstunde durchzuarbeiten. Außerdem anstatt sich wie die restlichen Team Mitglieder bei Problemen sich über diese Probleme zu informieren und bei nicht Verständnis sich an den Team Leader zu wenden fand es Leon Wilde adäquat vermehrt auf Anweisung dies nicht zu tun ans Handy zu gehen und nichts zu erreichen. Im Falle, dass Herr Wilde sich tatsächlich mit den Problemen sich befasst hatte musste der Team Leader feststellen, dass einige Code Teile von Ihm Plagiate oder eindeutige Kopien aus generativen Künstlichen Intelligenzen, wie [ChatGPT](https://chatgpt.com) zu sein scheinen. Die Aufgaben wurden außerdem zu keinem Zeitpunkt ausreichend bearbeitet was andere Team Mitglieder dazu zwang diese Aufgaben aufzuarbeiten.
    - Maurice Severin erhielt, die Aufgabe von dem Team Leader erhielt mit Leon Wilde seine Aufgaben erweiterten zu besprechen, da der Team Leader nicht es für möglich sah die Aufgaben ausführlich genug für Herrn Wilde zu besprechen ohne, dass er das restliche Team gefährden hätte
* **Maurice Severin:**
  + **Aufgaben:** Die Aufgabenverteilung ist identisch zu den Aufgaben von Leon Wilde. Zusätzlich hat er das Login System erstellt.
  + **Bearbeitung dieser Aufgaben** **(Bewertung durch Team Lead)**:
    - Während den Bearbeitungsphasen hatte er ordentlich seine Aufgaben bearbeitet und durch seine bereits drei Jährige Expertise sich außerordentlich eingebracht Private gegenüber dem Team Leader. Jedoch ist zu sagen, dass zu großen Teilen der gegeben Zeit Maurice Severin fehlend war.

### API Dokumentation

#### Wie spricht man die API an?

Die API ist lokalisiert im *„$root$/routes/api/api.php“*, „*$root$*“ hierbei ist der oberste Ordner worin auch die erste „*index.html*“ enthalten ist.

Grundlegend ist die API immer mit ähnelndem Schema anzusprechen. Die API muss immer mit der [POST HTTP Request Method](https://en.wikipedia.org/wiki/POST_(HTTP)) angesprochen werden. Es wird immer ein Parameter „*method*“ benötigen dieser Parameter sieht auch immer ähnlich aus. Als erstes wird immer eine Aktion kommen, bei der jetzigen Ausführung gibt es die Aktionen „*get*“, „*set*“, „*delete*“, „*update*“, „*option*“. Darauf folgt immer einer der zusammenfassenden Begriffe, was im jetzigem Moment die Tabellenstruktur der Datenbank widerspiegelt, also „*mitarbeiter*“, „*kunde*“, usw. Anschließend wird einer von bestimmten Worten erwartet, Beispielsweise „*bykey*“, „*all*“, „*main*“. Damit kommt als Parameter „*method*“ beispielsweise der Wert „*getmitarbeiterall*“ herraus.

Sollte danach der Endpunkt weitere Parameter benötigen, dann wird dieser Endpunkt es in folgender Antwort Struktur dem vorherigem Skript mitteilen.

#### Antwortstruktur

Die API antwortet immer im [Content Type JSON](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type#content-type_in_a_rest_api_using_json) mit diesem Teilen, „*status*“, „*msg*“, „*data*“, „*code*“. Im „*status*“ werden immer entweder „*error*“ für einen Fehler drinnen stehen oder „*success*“ sofern, dass Skript so ausgeführt ist, wie es gemeint wurde. Die „*msg*“ wird immer eine in Englisch geschriebene Generische Nachricht enthalten, die auch dem User angezeigt werden kann. „*data*“ enthält immer die automatisch erstellte Nachricht oder nichts. Zum Schluss sollte „*code*“ den korrekten [HTTP Response Code](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status) enthalten.

#### Wo findet man Hilfe?

Zum einen hat die API eine eingebaute Funktion, um sich diese Endpunkte genauer anzeigen zulassen, indem man die gefragte „method“ mit „option“ prefixed. Anderen falls geht es auch sich eine PDF herunterzuladen worin alle Endpunkte definiert sind.

### Verwendeter Technologie Stack

Das Ticktierungsystem basiert auf einem LAMP (Linux, Apache, MySQL, PHP) System mit Verwendung von Vanilla Javascript im Frontend und PHP im Backend. Die Verwendeten Libaries liegen Lokal auf dem Server und sind zum einem JQuery/AJax und JS-PDF mit dem Autotable Plugin zur Generierung von PDF‘s. Und es nutzt Cookies für die Session Speicherung. Ansonsten nutzt das Projekt keine weiteren Technologien.

Zukünftig ist anzustreben jetzt wie die grundlegenden Funktionen einmal ausgeschrieben geworden sind zu einem Framework wie [Svelte](https://svelte.dev/) oder [NextJS](https://nextjs.org/) zu wechseln, um eine vereinfachte Form fürs Frontend Development zu erstellen. Außerdem sollte man in Erwähgung ziehen die API auf eine performantere Sprache umzuschreiben, da PHP in größeren Abfragen höhere Latenz mit sich zieht.

1. gegebenfalls [↑](#footnote-ref-2)
2. floating deadline – nicht festgelegtes Abschlussdatum [↑](#footnote-ref-3)
3. Admin Panel - Administrationsoberfläche [↑](#footnote-ref-4)