# 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](#_Toc1)

[Projekt der Oberstufe 1](#_Toc2)

[Abkürzungsverzeichnis 3](#_Toc3)

[1. Anforderungskatalog 4](#_Toc4)

[2. Entity Relationship Diagramm 6](#_Toc5)

[3. Relationales Datenbankmodell 7](#_Toc6)

[4. Physischer Datenbankentwurf 8](#_Toc7)

[Erstellung der Datenbank 9](#_Toc8)

[Erste Entwürfe 9](#_Toc9)

[Warum MySQL und nicht PostgreSQL? 9](#_Toc10)

[Verwendung der Webseite 10](#_Toc11)

[Erste Anmeldung 10](#_Toc12)

[Einrichtung eines weiteren Accounts 10](#_Toc13)

[Team-Aufgaben Verteilung 11](#_Toc14)

[Team Lead 11](#_Toc15)

[Frontend 11](#_Toc16)

[Backend 13](#_Toc17)

[Kreisdiagramm zur Aufgabenteilung im Projekt 14](#_Toc18)

[14](#_Toc19)

[API Dokumentation 15](#_Toc20)

[Wie spricht man die API an? 15](#_Toc21)

[Antwortstruktur 15](#_Toc22)

[Wo findet man Hilfe? 15](#_Toc23)

[Verwendeter Technologie Stack 16](#_Toc24)

[Fazit 17](#_Toc25)

[Fazit von Timon Wegner 17](#_Toc26)

[Fazit von Lia Kaup 18](#_Toc27)

[Fazit von Leon Wilde 18](#_Toc28)

[Fazit von Jadon Sill 19](#_Toc29)

[Fazit von Timo Szadkowski 20](#_Toc30)

[Fazit von Maurice Severin 20](#_Toc31)

## Abkürzungsverzeichnis

API - Programmierschnittstelle

MySQL – ein Open Source Projekt für eine Relationale Datenbank geleitet von Oracle

PostgreSQL – ein Open Source Projekt für eien Relationale Datenbank geleitet von der PostgreSQL Globaldevelopment Group

### 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:

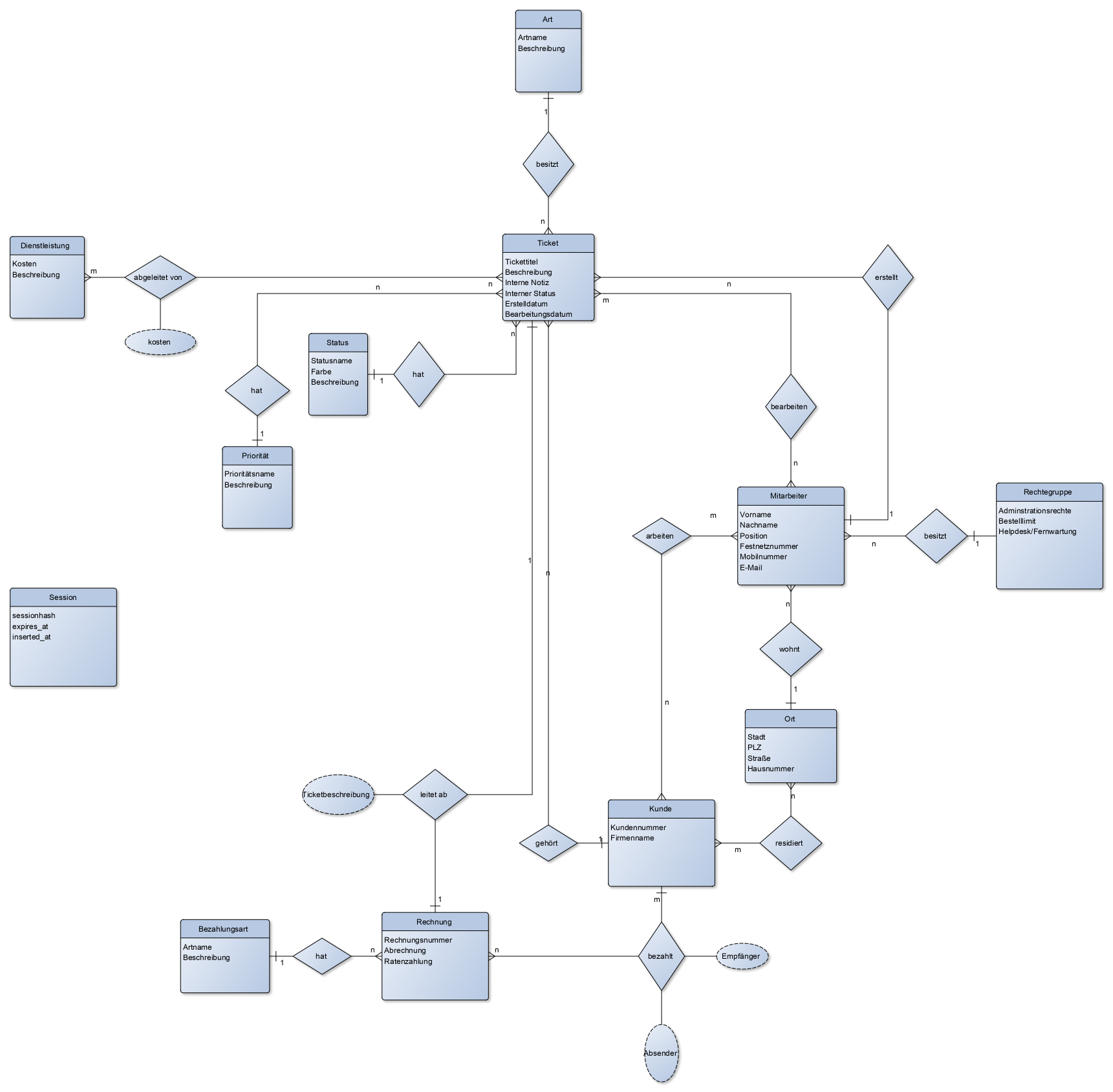
|  |  |  |
| --- | --- | --- |
| **Aufgabe** | **Beschreibung** | **Priorität** |
| **Ticket-Erstellung ermöglichen** | Benutzer müssen neue Tickets für verschiedene Anfragen oder Probleme erstellen können. | Hoch |
| **Statusverwaltung von Tickets** | Tickets sollten unterschiedliche Status (z. B. offen, 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

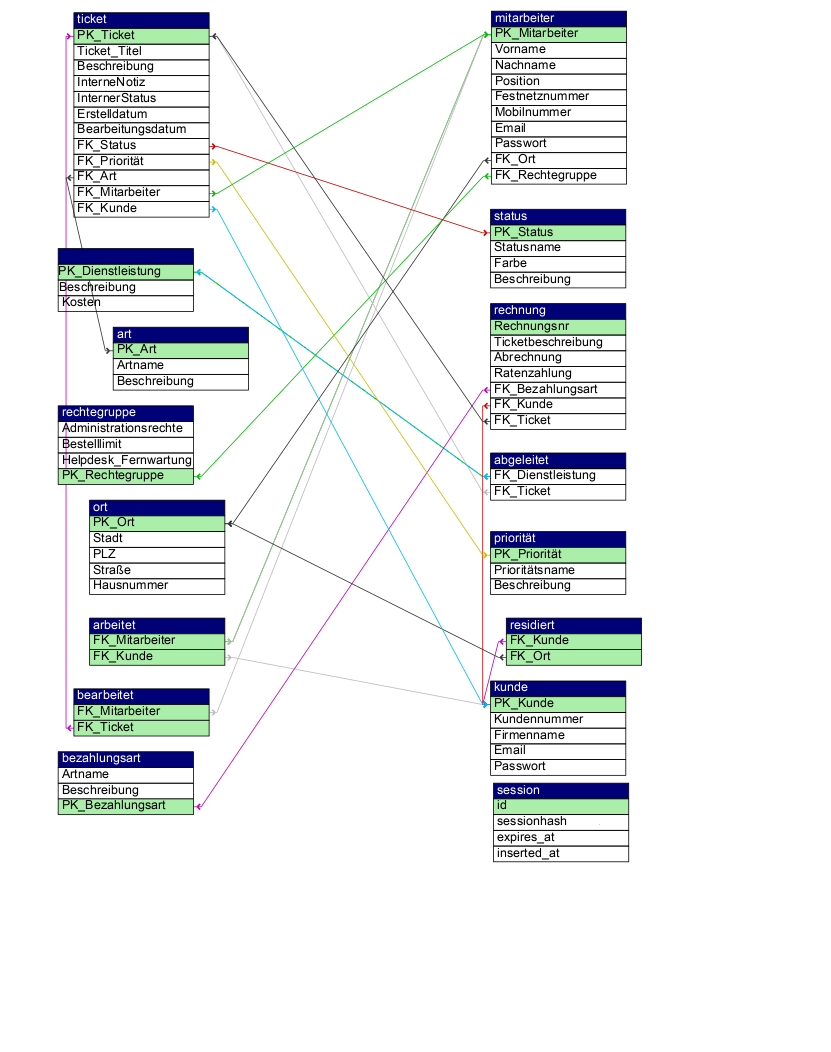
residiert: *FK\_Kunde, FK\_Ort*

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



### Erstellung der Datenbank

#### Erste Entwürfe

Die ersten Entwürfe der Datenbank sind ziemlich Deckungsgleich zu der Final erstellten Datenbank, der einzige Unterschiede zwischen der jetzigen und vorherigen Datenbank ist, dass die Tabelle für die Mitarbeiter für die Kunden mit verbunden wurde, damit haben wir jetzt zukünftig die Möglichkeit zu implementieren, dass Kunden auch ohne einen Mittelsmann der eigenen Firma, Tickets erstellen können und Rechnungen einsehen können.

Die Erstellung dieser Datenbank verlief chronologisch, man fang damit an die Grundlegenden Strukturen immer wieder in kleineren ER Modellen zu verfeinern und führte sie anschließend zusammen, darauffolgend erstellte man das Relationale Modell in Textform, um zum Schluss die Datenbank in [MySQL](#MySQL) zu schreiben.

#### Warum MySQL und nicht PostgreSQL?

Durch die Größe der Datenbank, wollten wir ein System wählen welches bereits eine Bandbreite an Support für alle gängigen Mittel erhalten hatte, beispielsweise ist der [PostgreSQL](#PostgreSQL) Support in der Sprache PHP ziemlich degradiert im Vergleich zu MySQL. Zum anderen ist die Replikation von PostgreSQL Datenbanken weiterhin ein Problem und sind in vielen Fällen nicht so umfangreich dokumentiert.

Uns ist aber bewusst, dass dies Zukünftig sich verbessern sollte und wir, dann sofern es dazu kommt das System umschreiben sollten, damit wir die in PostgreSQL bereits eingebauten JSON Funktionen verwenden können.

### 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 Navigationsleiste auf Login, um sich für das Administrations--Paneel anzumelden.

Darauffolgenden geben Sie die von uns gegeben Administrationsdaten ein, danach sind Sie in der Hauptseite des Adminstrations-Paneels und können von hieraus in der Navigationsleiste zu allen verschiedenen Unterpunkten gehen und dort mithilfe der Formulare immer wieder Daten eintragen. Hierbei sind bereits erste Demo Daten eingetragen.

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

#### Einrichtung eines weiteren Accounts

Um einen weiteren Account zu erstellen müssen, sie auf den Mitarbeiter Reiter im Administrations-Paneel gehen und dort einen neuen Benutzer anlegen, sollte dieser nur ein Kunde seien suchen und wählen sie den Kunden herraus. Sollte dieser auch Administrationsrechte auf diese Applikation bekommen, so wählen sie ihre eigene Firma aus. Eine jetzige Restriktion ist, dass die eigene Firma immer die erste Firma seien muss welche in der Datenbank eingetragen ist.

### 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 fast aller 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.

#### Kreisdiagramm zur Aufgabenteilung im Projekt

#### 

### 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.

Die Auswahl dieser Technologien wurden zum einem getroffen, da das Team bereits vertraut mit diesen Lösungen ist, aber auch damit das Team extremst schnell und mit wenig Abhängigkeiten den Prototypen erstellen und editieren können.

Während den Bearbeitungsphasen sind wir erneuter Weise immer auf die Limitierungen von JS-PDF getroffen, hierbei ist es von uns empfehlenswert möglicherweise ein Programm zu entwickeln worin dies für diese Applikation besser vonstatten geht.

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 Erwägung ziehen die API auf eine performantere Sprache umzuschreiben, da PHP in größeren Abfragen höhere Latenz mit sich zieht.

### Fazit

Im folgenden sind die Fazite der Gruppe, diese sind nicht editiert sofern keine unerhörte Sprache verwendet wurde. Sollte dies geschehen sein, wird in dies gekennzeichnet und Zensiert.

#### Fazit von Timon Wegner

Im Allgemeinen ist das Projekt ziemlich gut gelungen, in Zukunft sollte man jedoch auf ein mehr geleitetes Projekt zurück greifen, da anderenfalls es immer ein bisschen schwierig ist für einige, dass Projekt zu strukturieren. Ich habe aus mehreren anderen Gruppen mitbekommen, dass die Phase nicht besonders gut funktioniert hat. Aus meiner Sicht ist es zwar die richtige Wahl die Autonomie den Schülern zu überlassen, da es in der realen Welt nicht anderweitig abläuft, aber leider mit den wenig eingeschränkten Regeln ist es mir aufgefallen, dass einige sich zu viel aufgenommen haben.

Die Gruppenatmosphäre war zu meinen Erkenntnissen mehr als konstruktiv und ausreichend, um eine fördernde Lernatmosphäre zu bilden auch, wenn nicht alle Gruppenmitglieder, dies ausnutzen zu schienen.

Zusätzlich ist die Infrastruktur nicht für solch welche Projekte ausgelegt, weder die Installation von Version Control wie „git“, noch die Möglichkeit auf einen Zentralen Server von dem alle Arbeiten können besteht, dies hat bei unserer Gruppe Frühzeitig abfinden gefunden, da wir uns von dem Lehrer Herr Klitzke einen FTP-Zugang, sowie ein Platz auf seinem Webserver geben lassen haben, dies sollte jedoch nicht die Lösung sein.

Ganzen im Ganzen sollten einige Dinge sich ändern sollen, damit solch eine Gruppenarbeit mehr an der realen Welt teilnimmt. Ich kann verstehen, dass das Einrichten eines Servers wahrscheinlich nicht mit den Mitteln der Schule möglich ist, man sollte aber in Erwägung ziehen vielleicht ein extra Netzlaufwerk für den schnelleren Austausch unter Schülern anzulegen wo nur die Gruppenmitglieder Zugriff drauf haben, dies existiert nämlich im jetzigem Moment nicht.

#### Fazit von Lia Kaup

Was hat mir gefallen:

Mir hat gefallen die Hilfsbereitschaft und die Geduld die Timon hatte, auch wenn ich nicht die hellste beim Programmieren bin. Timon konnte mir erklären in wenigen schritten wie ich was programmieren soll und hat mir auch bei Fehlern geholfen.

Die Gruppe generell war sehr hilfreich und hat hart gearbeitet, aber auch mal kleine Späße gemacht.

Was hat mir nicht gefallen:

Mir hat nicht gefallen, dass ich gerne mal abgelegt werde. Ich probiere selber mich auf meine arbeiten zu konzentrieren, aber selbst da werde ich mal gerne von meinen Arbeiten abgelenkt

Was möchte ich verbessern:

Ich möchte an mir verbessern, weniger am Handy zu sein auch wenn es manchmal sehr verlockend ist dran zu gehen. Auch möchte ich mich mehr trauen einfach fragen zu stellen und nicht eine halbe Stunde sitzen zu bleiben und nichts tun. Zum Schluss will ich so machen, dass ich nicht mehr so schnell abgelenkt werde, damit ich eher mit meinen Aufgaben fertig werde und es nicht zu lange braucht bis mein Teil fertig ist.

#### Fazit von Leon Wilde

Ich habe zur ganzen Arbeit kein Wirkliches Feedback. Wie das bei mir aussieht sieht man ja in Timons text was ich anderweitig natürlich nur widerspiegeln kann. Mit der Orga von ihm zu den anderen hatte ich nicht wirklich viel zu tun und sonst war das Projekt all in all ok.

#### Fazit von Jadon Sill

**Unterstützung durch Timon:**  
Bei Fragen oder Problemen konnte man sich jederzeit an Timon wenden. Er stand immer bereit, um zu helfen und unterstützte uns bei der Lösung von Schwierigkeiten.

**Aufgabenstellung und Lernmöglichkeiten:**  
Die gestellten Aufgaben waren klar und gut strukturiert. Sie ermöglichten es, neue Fähigkeiten zu erlernen und sich intensiv mit dem Thema auseinanderzusetzen.

**Organisation des Projekts:**  
Die Organisation des Projekts war sehr gut. Zu Beginn jeder Stunde wurde der Tagesplan klar besprochen, sodass alle wussten, was an diesem Tag zu tun war und welche Ziele verfolgt wurden.

**Verständlichkeit der Arbeitsprozesse:**  
Die Verständlichkeit, besonders in Bezug auf die Verknüpfung der Code-Teile zwischen den Teammitgliedern, war hervorragend. Es war einfach, die verschiedenen Teile des Codes zu integrieren und zu verstehen, wie sie zusammenarbeiten.

**Teamarbeit und Kommunikation:**  
Der Zusammenhalt im Team war stark. Besprechungen zu Meinungen und Designfragen verliefen konstruktiv und effektiv, was den Arbeitsprozess deutlich erleichterte.

#### Fazit von Timo Szadkowski

**Unterstützung durch Timon den Heiligen:**

Wenn es Fragen oder Probleme gab, konnte man sich an Timon wenden, der einem zur Verfügung stand und geholfen hat, eine Lösung zu finden.

**Aufgabenstellung und Lernmöglichkeiten:**

Die Aufgaben, die Timon uns gegeben hat, waren klar und gut zu verstehen. Dadurch konnte ich mein Wissen in Javascript und HTML erweitern.

**Organisation des Projekts:**

Die Organisation war gut. Am Anfang jeder Stunde wurde gesagt, wer was macht, sodass jeder wusste was zu tun ist.

**Verständlichkeit der Arbeitsprozesse:**

Die Verständlichkeit war hervorragend. Es war anfangs nicht leicht zu verstehen, wie die Teile des Codes zusammenarbeiten, aber am Ende war es dann klar.

**Teamarbeit und Kommunikation:**

Der Zusammenhalt im Team war gut. Besprechungen zu Meinungen und Designfragen verliefen konstruktiv und effektiv.

#### Fazit von Maurice Severin

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