Elektronikschule Tettnang

Klasse EFI221

Projektarbeit

Konzeption und Entwicklung einer Datenbank gestützten Webanwendung für die „Elektronikschule Tettnang“ Im Namen der „IT Solutions TT GmbH“

Verfasst von:

Tobias Blättlinger,
Keno Lässig

Betreuer: Herr Kompa, Herr Kopf, und Herr Giesen

Abgabetermin: 12.02.2024

Inhalt

1. Ausgangssituation

In der Elektronikschule Tettnang fällt bei der Wartung von den Rechnern in den Laborräumen ein unnötig großer Verwaltungsaufwand an, da der Lehrer bei Problemen mit den Arbeitsplätzen immer eine Mail mit der Problembeschreibung an die zentrale Netzgruppe der Schule schickt, welche die Aufgabe dann dem zuständigen Raumbetreuer weiterleitet. In Zukunft soll ein digitales Ticketsystem Abhilfe schaffen, um diesen Prozess zu vereinfachen. Als Azubis der IT Solutions TT GmbH sollen wir das Projekt leiten und umsetzen und stehen dabei im Austausch mit den Ansprechpersonen der Geschäftsleitung Herr Kopf und Herr Kompa.

1.1 Projektziele

Sachziel:

Es soll eine funktionsfähige Webanwendung entstehen, mit welcher eine Lehrkraft neue Tickets mit verschiedenen Daten anlegen kann. Die Lehrkraft sollte alle von selbst erstellten Tickets abrufen können und einzelne Details dieser sich ansehen können. Die Daten hierzu werden bei Betätigung eines Buttons an den Server geschickt und dort in einer Datenbank gespeichert.

Die Raumbetreuer sollten alle Tickets angezeigt bekommen, welche die Räume behandeln, für welche sie Zuständig sind. Es soll ihnen möglich sein, die Tickets zu bearbeiten und ihren Status zu ändern.

Die Tickets sollten nach dem Erstellungszeitpunkt sortiert angezeigt werden.

Terminziel:

Der letzte mögliche Termin einer Abnahme war am 25.02.2024 um 16 Uhr.

1.2 Teilaufgaben

Projektmanagement:

Es sollte ein Terminplan in Form eines Gantt-Diagramms erstellt werden. Außerdem die Ausarbeitung eines Personalplans, eines Kostenplans und einer Sachmittelplanung.

Teilaufgabe 1: Konzeption, Planung und Einrichtung der Datenbank

Erstellen des ER-Modells, Anlegen der Datentabellen, Eintragung der Primär– und Fremdschlüssel. Ein ER-Modell muss entworfen werden. Dabei sollen Primär– und Fremdschlüssel angegeben und Redundanzen vermieden werden. Dieses soll dann auf dem eingerichteten Datenbankserver erzeugt werden.

Teilaufgabe 2: Webprogrammierung

Es soll eine funktionsfähige Webanwendung programmiert werden, welche jedoch nur eine Seite umfasst. Auf der Seite soll ein „Login und Registrierung“-Pop-up erscheinen, nach welchem die jeweiligen Tickets angezeigt werden sollen. Des Weiteren soll ein Button zum Erstellen neuer Tickets existieren, welcher ebenfalls zu einem Formular Pop-up führt, in

welchem man dem Ticket seine Metadaten, wie einen Titel, die Beschreibung des Problems und einen Raum zuweist.

Teilaufgabe 3: Backendentwicklung

Es soll eine funktionsfähige Datenbankverbindung programmiert werden. Es sollen für Tickets und User eigene Klassen erstellt werden, welche zur Kommunikation mit der Datenbank als „Schablone“ dienen

1.3 Projektumfeld, Prozessschnittstellen, Projektressourcen

Projektumfeld:

Das Projekt wurde in einem Klassenzimmer der Elektronikschule Tettnang vom 23.01.2024 bis zum 25.01.2024 realisiert. Gearbeitet wurde in selbstgewählten 2er- Teams. In jedem Team wurde ein Teamsprecher ernannt, dieser leitete die tägliche Besprechung. Auch die Teams untereinander tauschten sich aus.

Prozessschnittstellen:

Als Ansprechpartner für Fragen oder Unklarheiten standen für die Dauer des Projekts die Lehrkräfte Giesen, Kompa und Kopf zur Verfügung, die gleichzeitig auch die Rolle der Geschäftsleitung einnahmen. Die Teammitglieder untereinander tauschten sich in den täglichen Besprechungen aus, sowie während des Arbeitstages.

Projektressourcen:

Ein Laborraum der EST mit Desktop-PCs stand für die Projektdauer zur Verfügung. Genutzt wurden eigene Firmenlaptops.

2. Ressourcen- und Ablaufplanung

1.1 Terminplanung

Um einen reibungslosen Ablauf zu ermöglichen, war es wichtig im Vorhinein eine genaue Terminplanung zu erstellen. Diese erfolgte mithilfe der Tools der Webseite draw.io. Die vier Projektphasen Definitions-, Planungs-, Durchführungs- und Abschlussphase bildeten den roten Faden des Projekts. Bei einigen Aufgaben konnte man wie in Abb. 01 eine parallele Bearbeitung einplanen.

Terminplan Ticketsystem			Dienstag (08:00 bis 16:00 Uhr)				
Aufgabe	Start	Dauer	8	9	10	11	12
Gesamtprojekt	23.01.2024 -08:00Uhr	72h					
1. Definitionsphase	23.01.2024 -08:00Uhr	1h					
Brainstorming	23.01.2024 -08:00Uhr	30min					
Rollenverteilung	23.01.2024 -08:30Uhr	30min					
2. Planungsphase	23.01.2024 -09:00Uhr	5h					
Terminplanung	23.01.2024 -09:00Uhr	1h					
Qualitätsplanung	23.01.2024 -10:00Uhr	1h					
Sachmittelplanung	23.01.2024 -11:00Uhr	1h					
Personalplanung	23.01.2024 -12:00Uhr	1h					
Kostenplanung	23.01.2024 -12:00Uhr	1h					

Abbildung 1

Bei anderen war Teamarbeit unbedingt erforderlich. Wichtige Meilensteine waren der Abschluss der Planungsphase mit Puffer am 23.01.2024 um 14:30 Uhr. Außerdem die Aufsetzung des Webservers bis 16 Uhr.

2.1 Personalplanung

In der ersten gemeinsamen Besprechung wurde sich auf die folgende Personalplanung geeinigt: Herr Kompa und Herr Kopf fungieren als Berater. Herr Blättlinger bereitet den Web- sowie den Datenbankserver und das Backend der Webseite vor. Außerdem entwirft er das Design der Datenbankstruktur und implementiert diese anschließend. Derweil kümmert sich Herr Lässig um die Implementierung des Front-Ends der Webseite. Anschließend führen Herr Blättlinger und Herr Lässig ihre Arbeit zusammen und kümmern sich um die Implementierung des Back-Ends.

2.2 Sachmittelplanung

Zur Durchführung des Projektes werden spezifische Sachmittel benötigt. Hier eine Auflistung aller benötigten Sachmittel:

Hardware:

Tisch 1x

Maus 1x

Stuhl 2x

Laptop 2x

Software:

XAMPP (Datenbank-, Webserver)

PHPMyAdmin (Datenbank erstellen und verwalten)

Visual Studio Code (Entwicklungsumgebung, Implementierung der Webseite)

Microsoft Excel (Gantt-Diagramm)

Draw.io (ER-Modell)

Windows 10, 11 (Betriebssystem)

Spring Boot (Backend Framework)

2.4 Kostenplanung

Aus den oben genannten Sachmitteln und den entstandenen Entwicklungskosten, wie Gebäudemiete, Personalkosten und vieles mehr ergibt sich daher folgender Kostenplan:

	Einheit	Preis pro Einheit	Beschreibung der Leistung	Betrag in €
Materialkosten	1	1,37 €	Tobis Laptop (500,00€ abgeschrieben auf 3 Jahre) für 4 Tage	1,37 €
	1	3,17 €	Kenos Laptop (869,00€ abgeschrieben auf 3 Jahre) für 4 Tage	3,17 €
	2	0,08 €	Stühle (146,50€ abgeschrieben auf 5 Jahre) für 4 Tage	0,16 €
	2	0,85 €	Schreibtisch (389,00€ abgeschrieben auf 5 Jahre) für 4 Tage	1,71 €
			Summe:	6,41 €
Gebäudekosten	1	65,75 €	Kaltmiete Gebäude inkl. Nebenkosten auf 4 Tage	65,75 €
	1	256,30 €	Heizkosten auf 4 Tage	256,30 €
	1	25,64 €	Stromkosten auf 4 Tage	25,64 €
	1	1,91 €	Wasserkosten auf 4 Tage	1,91 €
	1	156,00 €	Reinigungskräfte für 4 Tage	156,00 €
			Summe:	505,60 €
Personalkosten	24	8,57 €	Tobias Brutto-Stundenlohn	205,68 €
	24	9,00 €	Keno Brutto-Stundenlohn	216,00 €
	24	4,00 €	Lehrkraft Brutto-Stundenlohn(ungefähr)/10 Gruppen	96,00 €
			Summe:	517,68 €
			Gesamtsumme:	**1.029,69 €**

Für die Gebäudekosten wird Folgendes angenommen:

Miete mit Nebenkosten: 6.000€

Heizölverbrauch: 12.000l/Jahr

Stromverbrauch: 6.500kW/Jahr

Wasserverbrauch: 87m³/Jahr

Reinigungskraft: 3h/Tag

2.5 Qualitätsplanung

Um ein reibungsloses Gelingen zu garantieren, muss vor Durchführung des Projekts ein Abnahmeprotokoll erstellt werden. Der Plan sollte sich an Anforderungen des Kunden orientieren. Ziel dieses Projektes ist es eine Webanwendung und Netzwerkarchitektur zu schaffen, die Unternehmensprozesse schnell und fehlerfrei abwickeln kann und den einzelnen Mitarbeiter in seiner Tätigkeit unterstützt. Daraus folgte dieses Abnahmeprotokoll zur Selbstprüfung und Einhaltung des Konzepts.

3. Durchführung- und Auftragsbearbeitung

3.1 Teilaufgabe 1: Planung, Konzeption und Einrichtung der Datenbankstruktur sowie die Einrichtung eines Datenbank-Webservers

Um den Anforderungen des Kunden gerecht zu werden und eine klare Datenstruktur zu erhalten, wurde eine Datenbank mit drei Tabellen erstellt: Eine Tabelle “User”, “Ticket” und “Rolle”. Die Tabelle “User” dient dazu, dem User eine Möglichkeit zu bieten, sich einen Account anlegen zu können und sich im Nachhinein mit diesem anzumelden. Mit der Tabelle “Ticket” werden Informationen über das Ticket gespeichert, um diese später auf der Webseite anzeigen zu können. Die letzte Tabelle “Rolle” ordnet dem User die Rolle “Lehrer” oder “Raumbetreuer” zu. Die “User”-Tabelle enthält folgende Informationen: Die ID als PK, der Vorname, der Nachname, die E-Mail-Adresse, das Kennwort und die Rolle. Mit den folgenden Informationen ist die ”Ticket”-Tabelle gefüllt: Die ID als PK, die Raumbezeichnung als FK der Tabelle “Raum”, der Titel, die Beschreibung, das Erstellungsdatum, den Status und die UserID als FK der Tabelle “User”. In der folgenden Abbildung sieht man die Datenbankstruktur in visueller Form als ER-Modell.

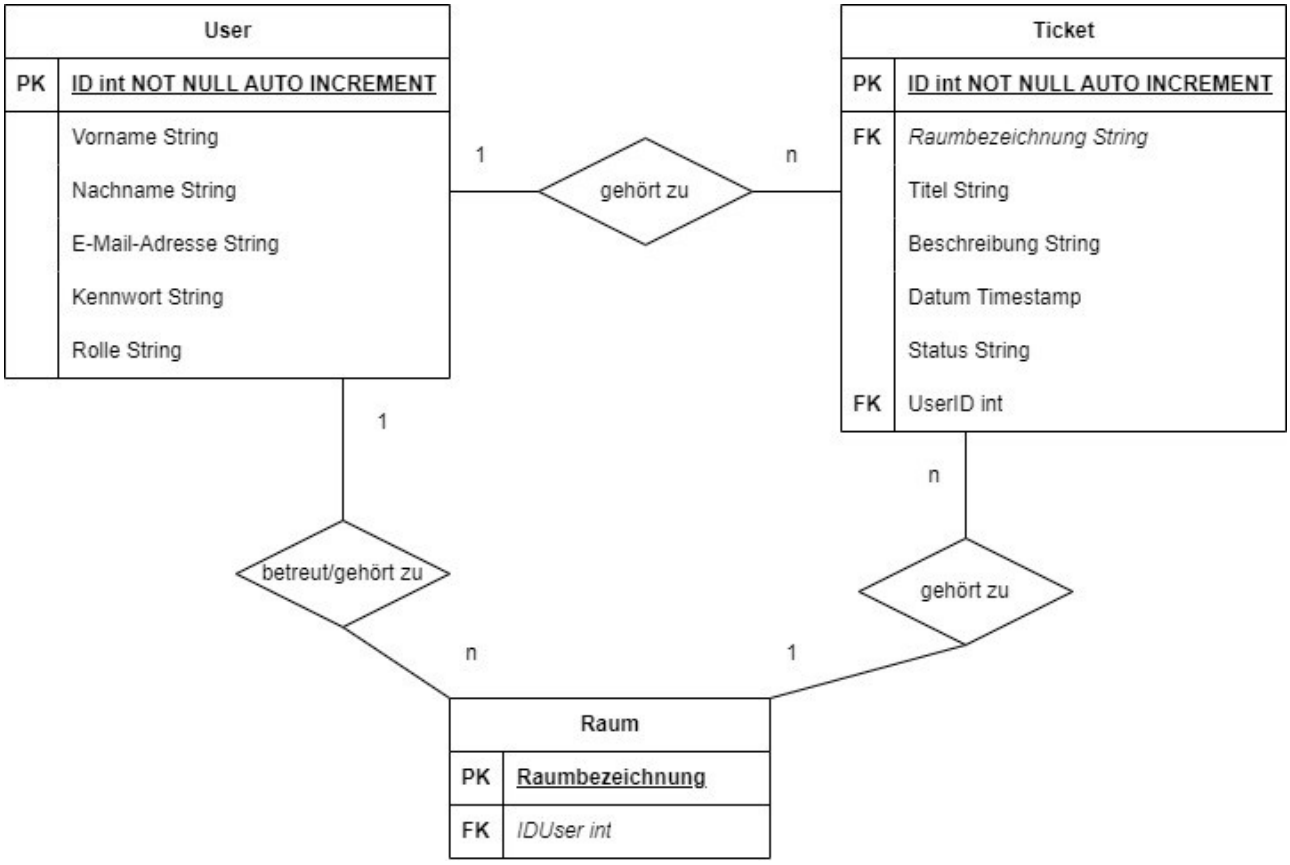


Abbildung 2

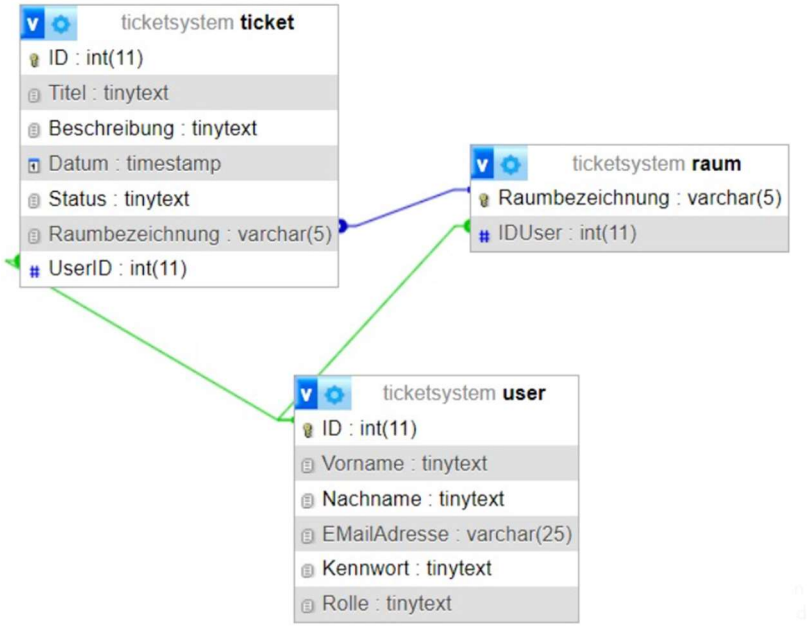


Abbildung 3

Die obige Abbildung zeigt die visuelle Darstellung der in PHPMyAdmin implementierten Datenbankstruktur mit den zu speichernden Informationen und Beziehungen zwischen den Tabellen.

Vorbereiten des Back-End Frameworks „Spring Boot“:

Vorbereitend für die Verbindung zwischen Webseite und Datenbank wurde ein Spring Boot Paket mit Hilfe eines Onlinegenerators erstellt. In der folgenden Abbildung sieht man die Webseite mit dem Generator, der man Spezifikationen beifügen kann. Die Abbildung zeigt die genauen Spezifikationen, die für das Projekt benötigt wurden:

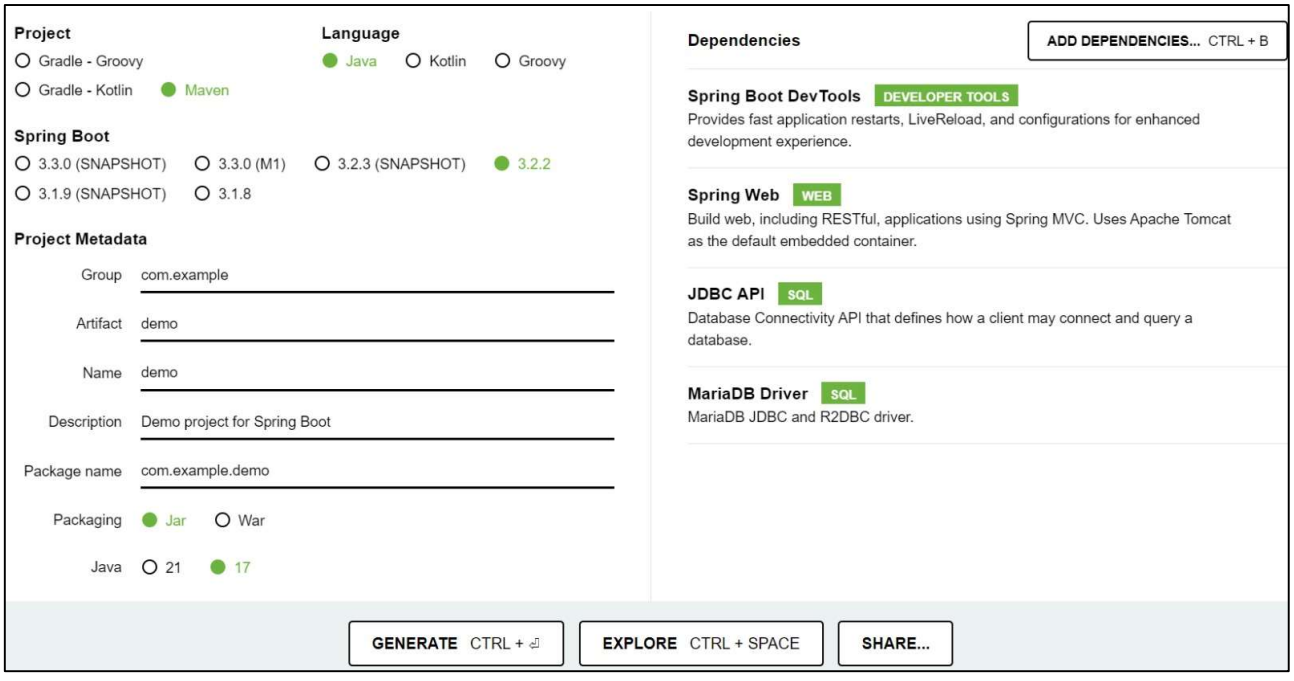


Abbildung 4

Nachdem Herr Blättlinger das Paket generieren lies, hat er alles aus der .zip Datei entpackt und in den “htdocs”-Ordner des “XAMPP”-Ordners übertragen, um dann später über den gestarteten Webserver auf die Seite zugreifen zu können.

Um nun eine Verbindung mit der Datenbank herstellen zu können, musste er in der Datei “application.properties” die URL, die zur Datenbank führt, eingeben sowie den Usernamen, das Passwort und den Treiber, den man verwendet, angeben.

```
spring.datasource.url=jdbc:mariadb://localhost:3306/ticketsystem
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
```

Zu guter Letzt hat er den Pfad “src\main\java” geöffnet und eine erste Testabfrage an die Datenbank geschrieben.

3.2 Teilaufgabe 2: Frontend Entwicklung

Das Framework:

Für die Frontend-Implementierung unseres IT-Projekts entschied man sich für das SAPUI5 Framework, welches mit XML-Views arbeitet, welche den zu sehenden Inhalt beschreiben und diesen beim Aufruf der View auf der HTML-Seite anzeigen, obwohl wir uns nicht in einer SAP-Umgebung befinden, aus den folgenden Gründen:

1. bietet SAPUI5 eine umfangreiche Sammlung von UI-Komponenten und -Bibliotheken, die es uns ermöglichen, eine moderne und ansprechende Benutzeroberfläche zu erstellen, die den Anforderungen unseres Projekts entspricht. Diese vorgefertigten Komponenten erleichtern die Entwicklung und beschleunigen den Prozess, indem sie bereits bewährte Lösungen für häufig auftretende Anforderungen bieten.

2. ist SAPUI5 ein Framework, das auf der OpenUI5-Bibliothek basiert, was bedeutet, dass es auf offenen Standards und bewährten Technologien wie HTML5, CSS3 und JavaScript aufbaut. Dies ermöglicht eine hohe Flexibilität und Erweiterbarkeit, da Entwicklerinnen und Entwickler ihre eigenen Anpassungen und Erweiterungen vornehmen können, um spezifische Anforderungen zu erfüllen.

3. bietet SAPUI5 eine enge Integration mit verschiedenen Backend-Systemen, nicht nur mit SAP-Lösungen. Obwohl wir uns nicht in einer SAP-Umgebung befinden, können wir dennoch von den Vorteilen der Integration mit verschiedenen Backend-Systemen profitieren, um eine nahtlose Kommunikation und Datenverarbeitung zwischen Frontend und Backend zu gewährleisten.

Insgesamt ermöglicht uns die Wahl von SAPUI5 für die Frontend-Implementierung eine effiziente Entwicklung, eine moderne Benutzererfahrung und eine nahtlose Integration mit verschiedenen Backend-Systemen.

Das Data-Model:

Der vorliegende Codeabschnitt dient dazu, Tickets von einem Server abzurufen und sie in einer Benutzeroberfläche anzuzeigen. Zunächst wird eine HTTP-Anfrage an den Server gesendet, um die Ticketdaten abzurufen. Sobald die Antwort des Servers empfangen wurde, werden die Daten in JSON-Format umgewandelt. Diese Daten werden dann in ein JSON-Modell geladen, um sie im Frontend zu speichern und zu verwalten. Das JSON-Modell ermöglicht eine effiziente Datenbindung und -verarbeitung in der Benutzeroberfläche, indem es eine klare Trennung zwischen Daten und Benutzeroberfläche schafft. Schließlich wird das erstellte Datenmodell der aktuellen Ansicht zugewiesen, damit die UI-Komponenten auf die Daten zugreifen und sie anzeigen können. Durch diese Vorgehensweise wird eine konsistente Datenverarbeitung innerhalb der Anwendung gewährleistet und eine nahtlose Integration von Backend-Daten in die Benutzeroberfläche ermöglicht.

```
fetch("http://localhost:8080/tickets")
        .then(function (response) {
                return response.json();
        })
        .then(function (data) {
                // JSONModel erstellen und Daten setzen
                var oModel = new JSONModel(data);
                that.getView().setModel(oModel);
        })
        .catch(function (error) {
                // Fehlerbehandlung
                console.error("Error fetching data: " + error.message);
        });
```

Ticketdarstellung

Für die Darstellung der Tickets haben wir uns für das SAPUI5-Element „GridList“ entschieden, da dies eine bereits passende Struktur und die nötigen Formatierungen mitbringt und dadurch recht schnell eine praktikable Darstellung ermöglicht. Die GridList unterteilt sich einzelne „GridListItems“, welche unsere einzelnen Tickets sein werden. Das machen wir möglich, durch das Definieren eines JSON-Pfades, wodurch die GridList das JSON-Model in einzelne Blöcke unterteilen kann.

```
<f:GridList
    id="gridList"
    mode="SingleSelectMaster"
    selectionChange="onTicketPress"
    growing="false"
    growingThreshold="9"
    items="{path: '/'}">
    <f:customLayout>
        <grid:GridBoxLayout />
    </f:customLayout>
    <f:GridListItem
    highlight="{
        path: 'status',
        formatter: '.formatStatus'
        }" >
        <VBox>
            <VBox class="sapUiSmallMargin">
                <layoutData>
                    <FlexItemData
                        growFactor="1"
                        shrinkFactor="0"
                </layoutData>
                <Title
                    text="Titel: {titel}"
                    wrapping="true" />
                <Label
                    text="ID: {id}"
                    wrapping="true" />
                <Label
                    text="Beschreibung: {beschreibung}"
                    wrapping="true"/>
                <Label
                    text="Status: {status}"
                    wrapping="true"/>
            </VBox>
        </VBox>
    </f:GridListItem>
</f:GridList>
```

In der Fertigen Applikation sieht das dann wie folgt aus: jedes Ticket steht mit seinen Details wie „{titel}“ oder „{beschreibung}“ in seiner eigenen Box, bildet also ein einzelnes „GridListItem“

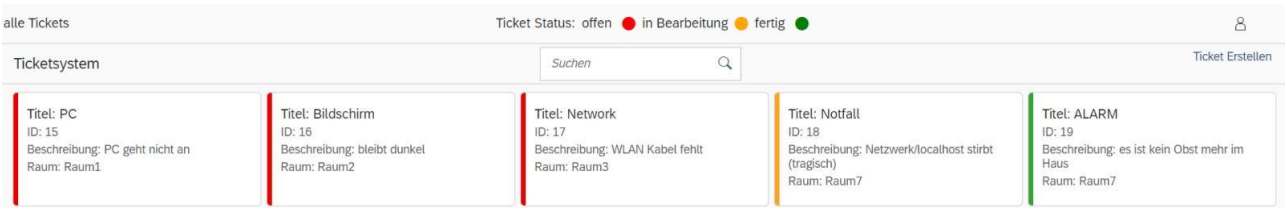


Abbildung 5

Für die darstellung von Dialogfenstern und Formularen, wie „Login“ oder „Registrieren entschieden wir uns für das SAPUI5-Modul „Fragmente“. Bei dieser Methodik erstellt man zusätzliche, kleiner XML-Dateien für die Pop-ups, welche man über Funktionen in der JavaScript jederzeit laden und auf der HTML anzeigen kann.

hier sehen wir ein solches Fragment für das Login-Formular:

```
<core:FragmentDefinition
        xmlns="sap.m"
        xmlns:core="sap.ui.core">
        <Dialog title="Login" contentWidth="40%" contentHeight="50%">
        <content>
        <Label text="Mail-Adresse" labelFor="mailInput" />
                <Input
                        id="mailInput"
                        type="Email"
                        placeholder="Enter mail-adress" />
      <Label text="Password" labelFor="passwordInput" />
                <Input
                        id="passwordInput"
                        type="Password"
                        placeholder="Enter password" />
      </content>
      <Toolbar height="30%">
         <Button
            text="Register"
            width="100px"
            press=".onRegisterButtonPress"
            class="sapUiSmallMarginBottom" />
                    <Button
            text="Login"
            width="75px"
            press=".onLoginButtonPress"
            class="sapUiSmallMarginBottom"/>
                    <Button
            text="close"
            width="75px"
            press=".onCancelButtonPress"
            class="sapUiSmallMarginBottom"/>
      </Toolbar>
   </Dialog>
</core:FragmentDefinition>
```

Und hier sehen wir den Aufruf des Fragments in der JavaScript Funktion:

```
this._dialog = sap.ui.xmlfragment("Ticketsystem.register", this);
this.getView().addDependent(this._dialog);

jQuery.sap.syncStyleClass(
        "sapUiSizeCompact",
        this.getView(),
        this._dialog
        );
this._dialog.open();
```

3.3 Teilaufgabe 3: Backend Entwicklung mit Java

SELECT-Befehl: ausgeben aller Tickets auf der Webseite

In der folgenden Abbildung sieht man ein Beispiel für einen unserer SELECT-Befehle. Hier werden alle Tickets aus der Datenbank selektiert und abgespeichert. Hierzu wird zunächst einmal eine Arraylist erstellt, um alle Tickets als eigene Entität abzuspeichern. Der nächste Schritt ist der Verbindungsaufbau zur Datenbank und das anschließende Ausführen des SELECT-Befehls. Nun werden die Einträge nacheinander durchlaufen und die Werte aus den Spalten in ein Ticket-Objekt geschrieben.

```
@GetMapping("/tickets")
    public String getTickets() {
        ArrayList<Ticket> tickets = new ArrayList<Ticket>();
        try (Connection con = jdbcTemplate.getDataSource().getConnection()) {
            String sql = "SELECT ID, Titel, Beschreibung, Datum, Status, Raumbezeichnung
                FROM ticket";
            PreparedStatement ps = con.prepareStatement(sql);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                Ticket ticket = new Ticket();
                tickets.add(ticket);
                ticket.setID(rs.getInt(1));
                ticket.setTitel(rs.getString(2));
                ticket.setBeschreibung(rs.getString(3));
                ticket.setDatum(rs.getDate(4));
                ticket.setStatus(rs.getString(5));
                ticket.setRaumbezeichnung(rs.getString(6));
            }
```

Um ein Ticket-Objekt erstellen und befüllen zu können, muss zunächst eine solche Klasse definiert werden, wie man in folgender Abbildung sehen kann:

```
public class Ticket {
    private int id;
    private String titel;
    private String beschreibung;
    private Date datum;
    private String status;
    private String raumbezeichnung;

  public int getID() {
        return id;
    }
  public void setID(int id) {
        this.id = id;
    }
            …
}
```

In dieser Klasse sind alle Informationen, die auch in der Datenbank abgespeichert werden, als Attribute hinterlegt und besitzen Setter- sowie Getter-Methoden.

Alle weiteren SELECT-Befehle sind auf dieselbe Weise aufgebaut.

In der folgenden Abbildung sieht man ein Beispiel für einen unserer INSERT-Befehle. Hier werden, nachdem sich ein neuer User registriert hat, alle Informationen in die Datenbank geschrieben. Hierfür wird zunächst eine Verbindung zur Datenbank hergestellt. Danach wird der INSERT-Befehl vorbereitet. Nun werden alle Informationen für die Registrierung in den Befehl eingebunden und anschließend wird der Befehl ausgeführt.

```
@PostMapping("/addUser")
        }
    }
        } catch (SQLException e) {
            e.printStackTrace();
            PreparedStatement ps = con.prepareStatement(sql);
            ps.executeUpdate();
            ps.setString(1, user.getVorname());
            ps.setString(2, user.getNachname());
            ps.setString(3, user.getEmailAdresse());
            ps.setString(4, user.getKennwort());
            ps.setString(5, user.getRolle());
    public String createUser(@RequestBody User user) {
        return "";
            String sql = "INSERT INTO user (Vorname, Nachname, EMailAdresse, Kennwort, Rolle)
VALUES (?,?,?,?,?)";
        try (Connection con = jdbcTemplate.getDataSource().getConnection()) {
```

Um sich nach der Registrierung anmelden zu können, muss wie beim Ticket auch für den User eine Klasse mit allen Attributen sowie Setter- und Getter-Methoden definiert werden.

3.4 Knackpunkte, Abweichungen, Anpassungen, Entscheidungen

Während der Bearbeitung des Projekts wurde das Team mit einigen Herausforderungen konfrontiert: Zum einen gab es mehrfach Probleme mit der Nomenklatur der Variablen in der Java(SpringBoot)- Komponente, welche das Übertragen von Paketen zwischen Datenbank und Frontend zeitweise fehlschlagen lies, des Weiteren war die geplante Struktur der personenbezogenen Inputs, z.B. „Registrieren“ zunächst nicht perfekt für einen „POST-Befehl“ abgestimmt war.

3.5 Qualitätssicherung

Anforderung	erfüllt	nicht erfüllt
Relationale Datenbank im ER-Modell erstellen:	X	
• mit Ticketdaten wie dem Status, Raum, Zuständigkeiten usw.	X	
• Raumdaten mit Zuständigem Raumbetreuer	X	
• Personaldaten mit Zugangsdaten und Beruf	X	
Datenbank anlegen	X	

Webanwendung mit:		
• Listendarstellung aller Tickets	X	
• Bei Klick auf ein Ticket ändert sich der Status		X
• Webformular zur Ticketerstellung, speichert die Angaben in der Datenbank ab	X	
• Sortierung der Tickets nach Erstellungszeitpunkt	X	
SpringBoot Entwicklung:		
• Ticket- und Userklassen	X	
• Select- und Updatebefehle	X	
Zusatzaufgaben:		
• Volltextsuche der Ticketliste	X	

Zugang mit Benutzernamen und Passwort	X	
Formular zum Erstellen von Tickets	X	

Fast alle der uns selbstgestellten Anforderungen konnten erfüllt werden. Leider konnte nur eine Zusatzaufgabe vollständig realisiert werden.

4. Projektergebnisse

4.1 Abnahme

Am 25.01.24 um etwa 15:30 Uhr wurde das Projekt von Herrn Kopf und Herrn Kompa abgenommen. Jeder Punkt wurde in der Anwendung und im Code überprüft. Bemängelt wurde die User-verifizierung im Frontend und mangelnde Funktion des Backends aufgrund fehlerhafter Variablenbenennung.

4.2 Soll-Ist-Vergleich

Aufgaben	Status	Geplante Zeit	Gebrauchte Zeit
Projektmanagement			
Projektauftrag	erfüllt		
Projektterminplan	erfüllt	1h	1h
Kostenplan	erfüllt	1h	1h
Qualitätsplan	erfüllt	1h	30min

Datenbank			
ER-Modell	erfüllt	1h	45min
Datenbankserver installieren	erfüllt	1h	15min
Tabellenstruktur erstellen	erfüllt	1h	1h
Webprogrammierung			
Backend Registrierung	erfüllt	20min	1h
Backend Login	nicht erfüllt	20min	1h
Backend Tickets anzeigen	erfüllt	20min	1h
Backend Tickets erstellen	teilweise erfüllt	20min	1h
Backend Ticketstatus ändern	erfüllt	20min	1h
Frontend Registrieren	teilweise erfüllt	1h	30min
Frontend Login	erfüllt	1h	30min
Frontend Tickets anzeigen	teilweise erfüllt	1h	1h
Frontend Tickets erstellen	erfüllt	1h	30min
Frontend Ticketstatus ändern	nicht erfüllt	1h	Plan wurde verworfen
Zusatzaufgabe			
Volltextsuche in der Ticketanzeige	erfüllt	2h	20min

4.3 Bewertung (Fazit, Ausblick)

Fazit:

Das Projekt konnte zufriedenstellend abgeschlossen werden. Alle Seiten der Webanwendung verrichten einem Prototyp ähnelnden Dienst, die Datenbank ist redundanzarm aufgebaut und ermöglicht einen einfachen, für Elektronikschule passenden Prozess. Kleine Teile des Projekts waren nicht komplett ohne Fehler, allerdings beeinträchtigten diese den Gesamtablauf nur rudimentär.

Ausblick:

Im Zuge auf das vom Geschäftsführer erwartete Wachstum des Autohauses in den kommenden Jahren lässt sich die bestehende Webanwendung leicht anpassen. Die Datenbankgröße sollte auch kein Problem für das jetzige System sein. In der bestehenden Netzwerkarchitektur sind von Seiten des Routers aktuell nur zehn Clients zugelassen. Bei einem größeren Mitarbeiterwachstum würde auch die Anzahl der Geräte steigen, und die Anzahl der aktuell zugelassenen nicht mehr ausreichen. Dies kann man jedoch mit wenig Mausklicks sofort auf Anfrage ändern, weshalb der Zukunft nichts im Wege steht.

II Verzeichnis der Arbeits-/Hilfsmittel

Software	Microsoft Word
	Microsoft Excel
	Draw.io
	Visual Studio Code
	Springboot
	XAMPP
	PHPMyAdmin
Hardware	2x Firmenlaptop

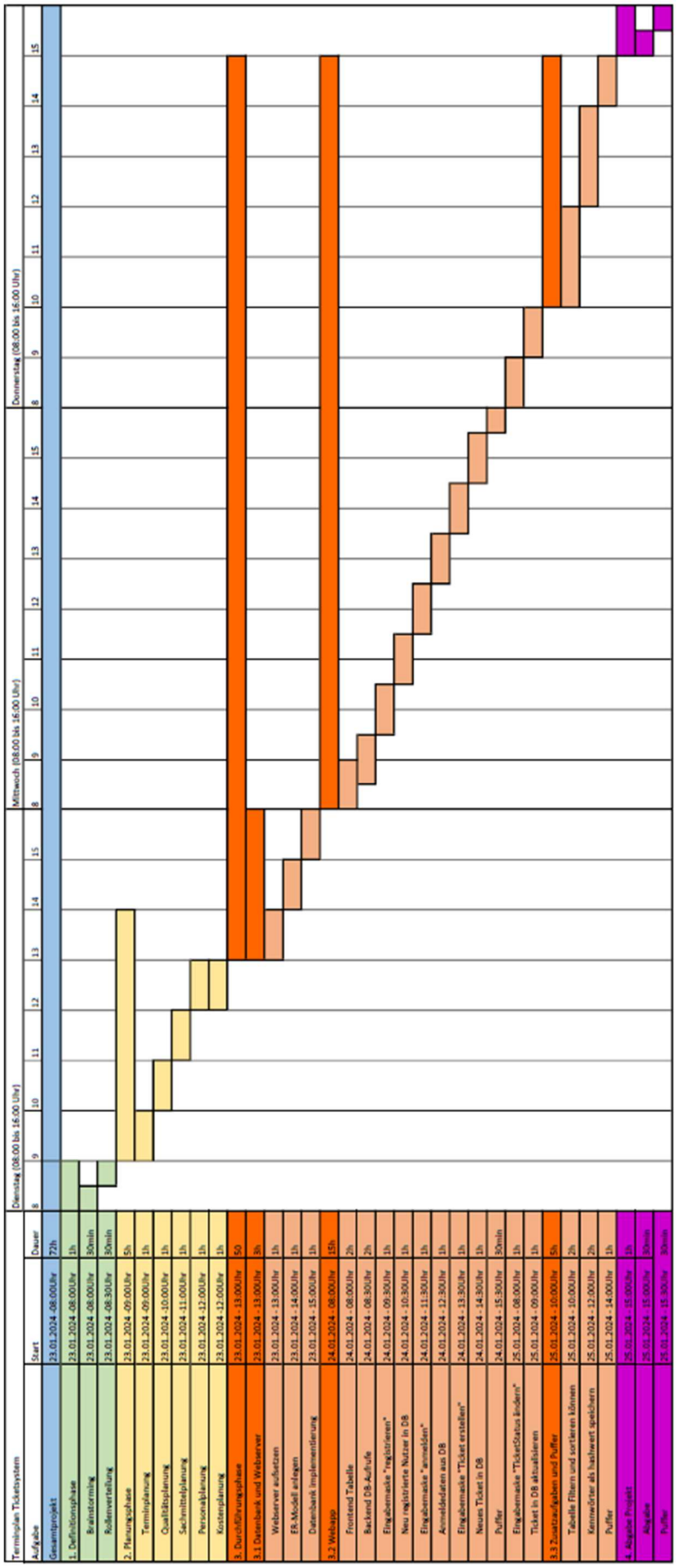
1. Abbildungsverzeichnis

Abkürzungsverzeichnis

HTML	Hypertext Markup Language
PHP	Hypertext Preprocessor
Abb.	Abbildung
SQL	Structured Query Language
UI	User Interface
ER	Entity-Relationship
SPA	Single Page Application
HTTP	Hypertext transfer protocol
HTTPS	Hypertext transfer protocol secure
CSV	Comma separated values
ID	Identifier
PK	Primary Key
FK	Foreign Key
URL	Unified Resource Locator
HTdocs	Hypertext Document

III Anhang

Anhang A: Terminplanung



IV Eidesstattliche Erklärung

Hiermit erklären wir, dass wie die vorliegende Arbeit selbstständig verfasst habe, dass wir sie zuvor an keiner anderen Hochschule und in keinem anderen Studiengang als Prüfungsleistung eingereicht habe und dass wir keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus Veröffentlichungen oder aus anderweitigen fremden Äußerungen entnommen wurden, sind als solche kenntlich gemacht.

Friedrichshafen, den 10.02.2024

Tobias Blättlinger

Hausen a.A, den 10.02.2024

Keno Lässig