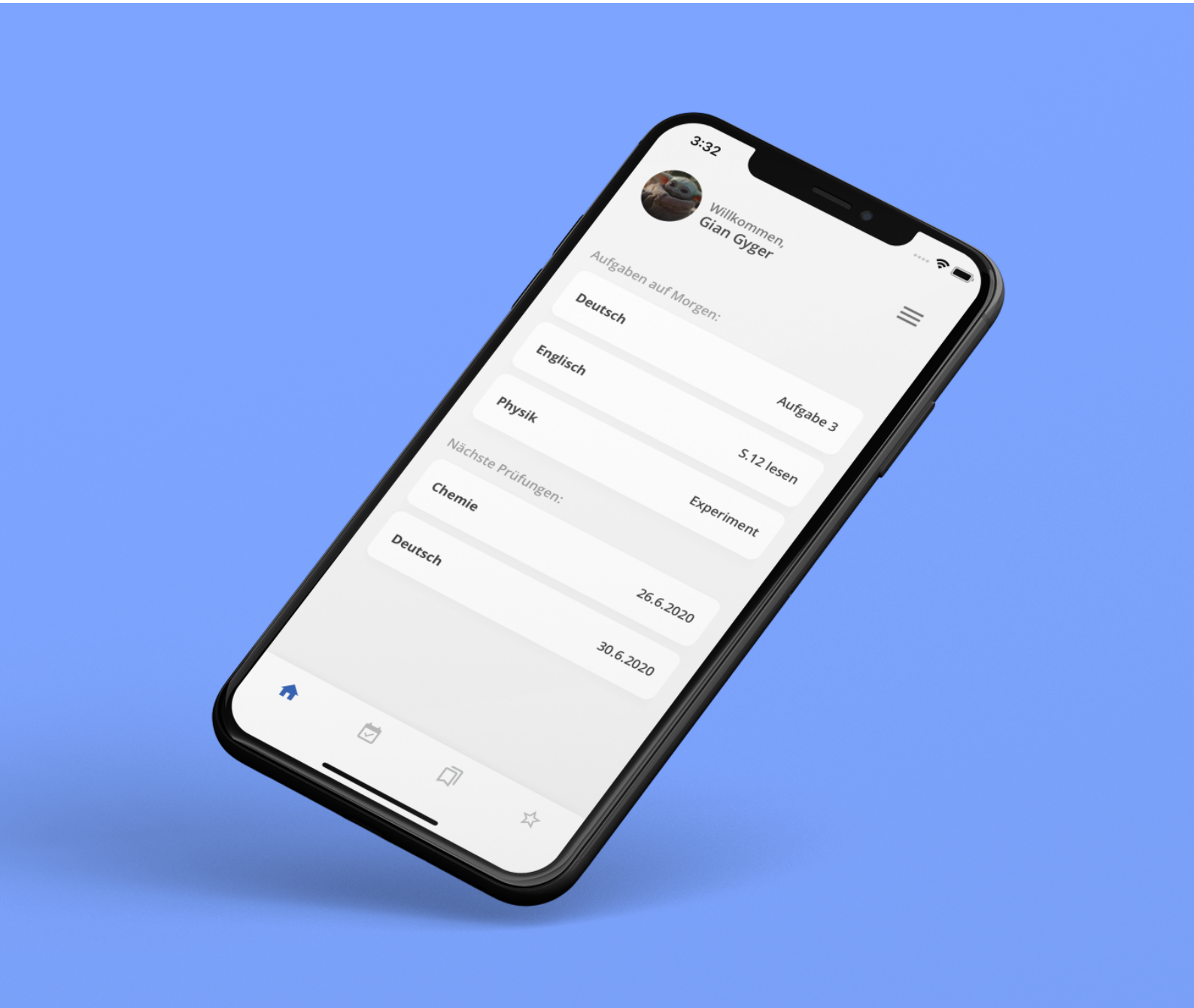
Kantonsschule Hohe Promenade, Gymnasium, Zürich
Maturitätsarbeit
Schuljahr 2020/2021

Proludo

- Entwicklung einer vollständigen App -

von Gian Gyger, Klasse 6a
Betreuer: Michael Liebich
Korreferent: Dr. Oliver Sieber



Inhaltsverzeichnis

1 Zusammenfassung

In dieser Arbeit geht es um die vollständige Entwicklung einer App für Schülerinnen und Schüler, die eine Agenda für Aufgaben und Prüfungen, eine Lernkartei zum Wörtchenlernen und eine Notenverwaltung beinhaltet. Die App ist als Web-App unter `https://proludo.net` uns als Mobile App im Apple App Store und im Google Play Store verfügbar. Sowohl die API als auch die Datenbank sind auf Cloud-Servern gehostet. Nach der Entwicklung wurde ausserdem in verschiedenen sozialen Netzwerken Werbung geschaltet, um neue Nutzer zu gewinnen.

2 Idee und Konzept

2.1 Idee

Die Idee von Proludo ist durch mehrere Probleme, die ich im Schulalltag hatte, entstanden und reicht bis ins erste Jahr am Gymnasium zurück. Zum einen war da die Schwierigkeit, die Übersicht über alle Schulnoten und Notenschnitte zu behalten. Ich nahm mir zur Lösung dieses Problems vor, eine App zur Notenverwaltung und automatischen Schnittberechnung zu entwickeln, um jederzeit den Überblick über meine schulischen Leistungen zu behalten. Das zweite Problem bestand darin, die Übersicht über Hausaufgaben und Prüfungen zu behalten. Die traditionelle Lösung dazu war natürlich sich eine Agenda zu besorgen, jedoch leben wir ja in der Zeit der Digitalisierung und eine App bietet mehrere entscheidende Vorteile: Man hat das Handy jederzeit dabei und kann somit auch unterwegs jederzeit einen kurzen Blick darauf werfen. Ausserdem kommt man in der App schneller zu seinen Informationen und hat einen besseren Überblick. Das dritte Problem war, Wörter zu lernen. Lehrmittel zu verwenden ist ungeeignet, da man immer durch alle Wörter durchgehen muss und die Reihenfolge sich dabei nicht verändert. Nahe liegt hier das Verwenden von Karteikarten, jedoch ist es äusserst aufwendig, diese zu schreiben. Sie brauchen viel Platz und sind schnell verloren. Eine App ist hier eine ausgezeichnete Lösung, da sie alle diese genannten Probleme behebt und ausserdem mit verschiedenen Lernmethoden die Möglichkeit abwechslungsreichen und effizienten Lernens bietet. Es sind also drei App-Ideen entstanden, die ich in einer App vereinen wollte. Diese App soll Schülern und Schülerinnen den Schulalltag mit diesen Funktionen vereinfachen:

- Agenda zur Verwaltung von Aufgaben und Prüfungen
- Lernkartei zum effizienten Lernen von Wörtern
- Notenverwaltung zur Übersicht über seine schulischen Leistungen

2.2 Konzept

Aus der Idee mit diesen 3 Hauptkomponenten entstand nun das Konzept einer kompletten Schülerplattform, die ich später Proludo (pro - für + ludus - Schule = für die Schule) nannte. Naheliegend war die Entwicklung einer Webapp und einer Mobile App für iOS und Android, die alle mit einer zentralen Schntittstelle in der Cloud verbunden sind, um dort die Daten zu verarbeiten und in eine Datenbank zu schreiben. So kann jederzeit und von allen Geräten auf die eigenen Daten zugegriffen werden, die weder durch Verlust des Gerätes noch versehentlichen Löschens der App verloren gehen können. Der Vorteil

einer Webapp ist, dass sie von jedem Gerät mit einer Internetverbindung erreichbar ist, ob dies nun ein Computer oder ein Handy ist. Dies ist allerdings gleichzeitig auch ein Nachteil, da die Webapp ohne Internetverbindung, zum Beispiel im Keller oder in der S-Bahn bei der Durchfahrt im Tunnel, nicht funktioniert. Ausserdem können bestimmte Funktionen des Gerätes nicht verwendet werden und die Performance der Applikation ist im Web stark eingeschränkt. Genau deswegen soll auch eine App für iOS und Android entstehen, die im AppStore und im PlayStore heruntergeladen werden kann. Die Vorteile dabei sind vor allem das Nutzererlebnis (User Experience), was ein sehr wichtiger Faktor für den Erfolg einer App ist, denn wenn die App sich nicht gut "anfühlt", ist sie schnell wieder deinstalliert. Ausserdem können so die Daten auch lokal auf dem Gerät gespeichert werden und die App kann offline verwendet werden. Der Vorteil am Hosten in der Cloud ist, dass keine Wartung nötig ist; diese wird vom Provider selbst durchgeführt. So muss man sich nicht darum kümmern, dass der Server 24/7 läuft und funktioniert. Ausserdem können die Server mit einem Knopfdruck automatisch skaliert werden, wenn die Leistung nicht mehr ausreicht.

2.3 Preismodell

Da ich von Anfang an im Hinterkopf hatte, ein kommerzielles Produkt zu entwickeln, musste ich mich für ein Preismodell entscheiden. Zum einen gibt es für Software das klassische Lizenzmodell: Der Nutzer muss, z.B. wie bei Windows, eine einmalige Zahlung tätigen und hat dann lebenslangen Zugriff auf die Software. Diese Variante ist für Proludo jedoch nicht geeignet, da die Infrastruktur, um die Software zu betreiben, gemietet wird und diese für den Entwickler monatliche und nicht einmalige Gebühren nach sich zieht. Der andere Typ ist das modernere Software-as-a-service-Modell (SaaS). Dabei wird dem Kunden monatlich oder jährlich ein geringer Betrag abgerechnet, und wenn dieser das Produkt nicht mehr verwenden will, kann er das Abonnement künden. Da mit diesem Modell ein monatliches Einkommen generiert wird und damit die Serverkosten bezahlt werden können, ist dieses Modell am besten geeignet. Ausserdem ist es für den Kunden attraktiver, einen geringen Betrag periodisch zu bezahlen als einmalig einen hohen Preis zahlen zu müssen. So lassen sich mehr Kunden generieren, was auch für mich lukrativer ist. Da es sehr schwierig ist, Kunden davon zu überzeugen, Geld zu zahlen, habe ich mich dazu entschlossen, eine kostenlose 30-tägige Testversion anzubieten, die alle Funktionen beinhaltet. Um die App danach weiter benutzen zu können, muss entweder das monatliche oder jährliche Abonnement ausgewählt werden.

3 Entwicklung

3.1 Planung

Bevor man ein Softwareentwicklungsprojekt startet, ist eine sorgfältige Planung der Entwicklungsschritte, die Festlegung der benötigten Technologien und der Softwarearchitektur notwendig. Von Anfang an wusste ich, dass ich eine Webapp, Mobile Apps für Android und iOS und ein cloudbasiertes Back-End wollte. Als erstes habe ich die dazu passenden Technologien ausgewählt und dann angefangen zu planen, wie die Entwicklung ablaufen soll. Ich habe einen Zeitplan entworfen, der ab Zeitpunkt des Beginns der Maturaarbeit gültig war. An diesem Punkt hatte ich bereits einen grundlegenden Prototypen entwickelt. Es hat sich aber schnell herausgestellt, dass dieser Plan viel zu ambitiös war und die Entwicklung einiges länger dauern würde.

Termin	Dauer	Inhalt
27.4.2020	4 Wochen	Web-App und Backend fertig mit allen Funktionen
8.6.2020	6 Wochen	Mobile App fertig mit allen Funktionen
15.6.2020	1 Woche	Performance-Optimierung
29.6.2020	2 Wochen	Feinschliff, Rechtliches, Veröffentlichung, Werbung

3.2 Architektur

Grundsätzlich braucht es für eine App drei verschiedene Komponenten: Eine Datenbank, um alle Daten und Nutzerinformationen zu speichern, einen Server für die Datenverarbeitung und das Front-End, welches der Nutzer zu Gesicht bekommt. Dies kann z.B. eine Webseite oder eine Mobile App sein. Da Proludo relativ komplex ist, beinhaltet die Softwareinfrastruktur einige Komponenten mehr. Proludo hat für die Datenbank einen

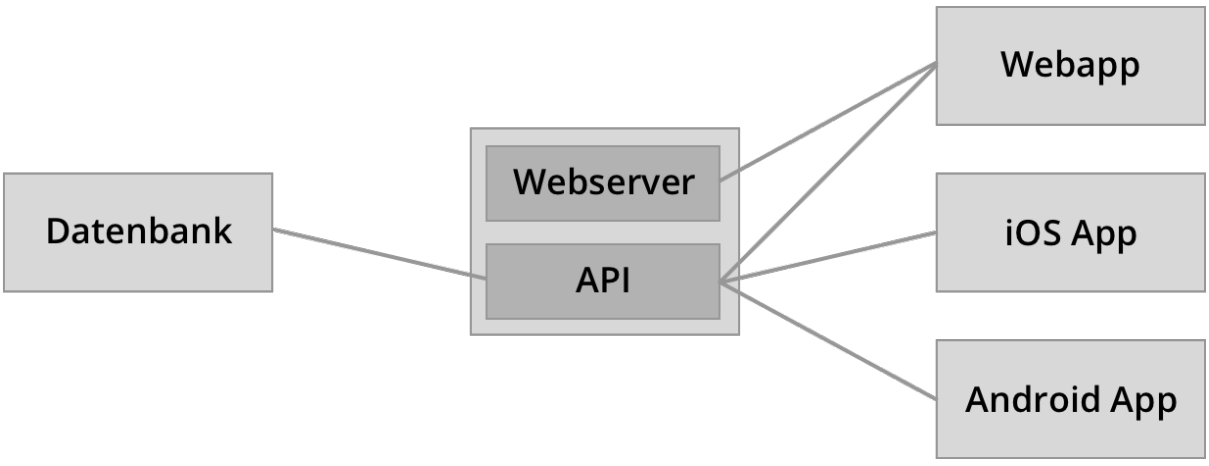


Abbildung 1: Softwarearchitektur

eigenen Server in der Cloud. Darin werden alle Nutzerdaten, also Benutzernamen, Email-Adressen, Passwörter und alle Daten wie Agenda-Einträge, Lernsets und Noten gespeichert. Diese Datenbank ist mit einem zweiten Server verbunden, auf dem zum einen die Webseite gehostet wird. Wenn also jemand die URL proludo.net aufruft, wird dieser Server angefragt und die Webseite von Proludo wird zurückgesendet. Gleichzeitig dient dieser

Server aber auch als API. Diese wird unter der URL `proludo.net/api` von der Webapp und den Mobile Apps aufgerufen und dient als zentrale Schnittstelle. Wenn zum Beispiel ein neues Lernset hinzugefügt, ein neuer Account erstellt oder eine bestimmte Information abgerufen wird, wird ein Request an diese Schnittstelle gemacht. Auf dem Server wird dieser Request dann bearbeitet, zum Beispiel werden Daten von der Datenbank angefordert und zurückgesendet oder ein neuer Eintrag wird in die Datenbank geschrieben. Die Webapp und die beiden Mobile Apps sind das, was der Nutzer zu Gesicht bekommt und wofür der meiste Code geschrieben wurde. Die Webapp wird vom Webserver beim Aufrufen der URL gesendet, die Mobile Apps können über die App Stores heruntergeladen werden. In diesen Apps stecken nämlich das ganze Layout und Design, aber auch die ganze Logik der App, wie zum Beispiel Schnittberechnungen, Karteisysteme und das Sortieren von Listen.

3.3 Technologien

Zur Entwicklung einer solch komplexen Plattform sind viele verschiedene Sprachen, Frameworks und Dienste nötig. Die Programmiersprache, die ich für die Programmierung hauptsächlich verwendet habe, ist JavaScript. Dazu kommen verschiedene Technologien, die entweder darauf aufbauen und viele weitere Funktionen bieten, oder die zur Ergänzung von Dingen, die mit JavaScript nicht möglich sind, benötigt werden. Die Sprache JavaScript habe ich zum einen ausgewählt, da sie für das Web die einzige Wahl ist, zum anderen weil sie enorm viele Erweiterungsmöglichkeiten durch Frameworks bietet. So konnte zum Beispiel die gesamte API mit derselben Sprache mit Hilfe von Node.js programmiert werden. Auch die Webapp wurde mithilfe von React mit JavaScript entwickelt. Und nicht zu vernachlässigen: Auch die Mobile Apps für Android und iOS, die normalerweise zwei verschiedene Programmiersprachen und komplett andere Tools und Bibliotheken benötigen, wurden mit dem Framework React Native und der Sprache JavaScript programmiert. Nun eine Übersicht über alle Technologien, die für die Entwicklung von Proludo verwendet wurden:

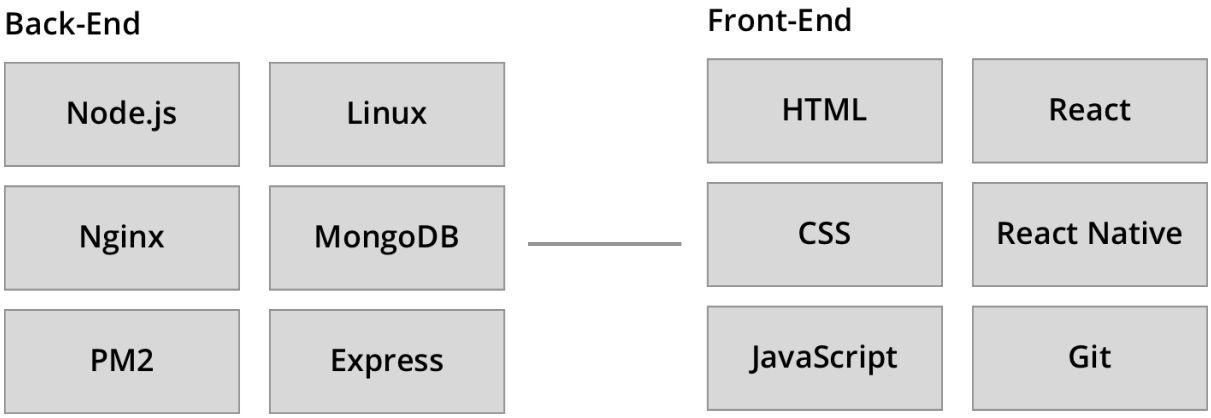


Abbildung 2: Verwendete Sprachen und Technologien

- **Node.js und Express:** Applikationen können dank der JavaScript Runtime Node.js nicht nur im Web, sondern auch auf Servern ausgeführt werden. Dies ermöglicht die Entwicklung von performanten Serveranwendungen. Mit dem Node.js-Framework Express können einfach Webserver und REST-APIs erstellt werden. Die gesamte API von Proludo wurde mit Node und dem Framework Express umgesetzt.

- **Nginx:** Diese Software läuft auf Linux-Servern und ermöglicht es u.a. Webserver zu betreiben. Proludo verwendet Nginx als API-Gateway, um alle Requests an die API weiterzuleiten und die Webseite komprimiert an den Client weiterzuleiten.
- **Linux:** Als Server-Betriebssystem wird Ubuntu Linux verwendet und in der Cloud in einem Container gehostet. Darauf läuft die gesamte Software, also API, Webserver und Datenbank.
- **MongoDB:** Diese Datenbank unterscheidet sich von anderen Datenbanken wie mySQL oder PostgreSQL dadurch, dass sie nicht auf der Sprache SQL basiert. Es ist eine sogenannte No-SQL oder Document based database. Dabei werden alle Informationen als JSON-ähnliche Dokumente abgespeichert. Dies ermöglicht mehr Freiheit und Performance beim Speichern von Daten.
- **PM2:** Dies ist eine Software, die auf dem Linuxserver läuft und für die Bereitstellung der API verantwortlich ist. Sie erlaubt eine einfache Skalierung durch Replikation der Server-Instanzen und Load-Balancing, dem Verteilen von Requests auf mehrere Instanzen, was der Performance und Sicherheit dient.
- **HTML:** Diese Sprache ist für den Aufbau und Inhalt einer Webseite zuständig. HTML steckt hinter jeder Webseite und dient zur Strukturierung der Webseiteninhalte.
- **CSS:** Diese Sprache wird zusammen mit HTML verwendet, um einer Webseite Styles bzw. ein Layout und Design zu verleihen. Mit dieser Sprache erstellt man Regeln, die dann auf bestimmte HTML-Elemente zutreffen. Solche Regeln sind z.B. Farbe, Schriftarten, Schriftgrössen und Abstand von anderen Elementen auf der Webseite.
- **JavaScript:** Der wahrscheinlich wichtigste Bestandteil von Proludo ist die Programmier-/Skriptsprache JavaScript. Sie ist unglaublich vielseitig und hat viele verschiedene Anwendungen. Am häufigsten wird sie auf Webseiten verwendet, um ihnen Funktionalität zu geben. Sie ist aber auch als Sprache für Mobile Apps gut geeignet. Ausserdem kann man sie auch auf Servern z.B. zur Programmierung von APIs verwenden. Bei Proludo erfüllt sie alle drei dieser Aufgaben.
- **React:** Dieses JavaScript-Framework ermöglicht die Entwicklung von komplexen Webapps, die vollständig auf JavaScript basieren. Mit Bausteinen, sogenannten Komponenten, lassen sich skalierbare Webapplikationen enwickeln.
- **React Native:** Wie React verwendet React Native die gleiche Syntax und basiert auf dem Komponenten-System und der Sprache JavaScript. Es ist ein Mobile Cross-Platform Framework, das heisst man schreibt nur einmal Code für Android und iOS. Im Vergleich zu anderen Mobile Frameworks wie Ionic verwendet React Native nicht einfach einen Web View, sondern die nativen Features des Betriebssystems. Dies ermöglicht eine fast so gute Performance, als würde man die App mit den systemeigenen Programmiersprachen schreiben, da Komponenten verwendet werden, die für das jeweilige Betriebssystem optimiert sind.
- **Git:** Git ist eine Software zum Versionsmanagement. Sie ermöglicht es, die ganze Code-Base zu verwalten und immer die Kontrolle über alle Versionen zu haben. Auch gegen Datenverlust oder Fehler ist man so gesichert.

- **Weiteres:** Als Authentifizierungsmethode werden JWT-Tokens verwendet. Es handelt bei einem Token um einen verschlüsselten String, der nach erfolgreichem Login serverseitig generiert wird. Dieser wird an den Client übermittelt, wo er als Cookie gespeichert wird. Dies ermöglicht den Login-Zugang, auch nachdem der Nutzer die Webseite oder App geschlossen hat, und vermeidet das Speichern vom Passwort und der E-Mail. Als Erweiterungen werden viele verschiedene Bibliotheken mit dem Package-Manager npm installiert, sowohl auf dem Server als auch in der Webapp und Mobile App. Zu diesen Bibliotheken gehören z.B. Redux, Redux Saga, Redux Offline, mongoose oder bcrypt. Mit bcrypt werden Passwörter gehasht. Das bedeutet, sie werden so verschlüsselt, dass man sie nicht mehr auf das Original zurückführen kann.

3.4 Services

Um Proludo und die damit verbundene Infrastruktur überhaupt betreiben zu können, sind verschiedene Services nötig. Dazu gehört zum Beispiel die Bereitstellung und das Betreiben von Server-Hardware, der Datenbank, Mailservices, Zahlungsabwicklung und vieles mehr. Es wird meist ein Service zur Bereitstellung einer bestimmter Technologie verwendet. So ist z.B der Service MongoDB nur für die Bereitstellung der Datenbank verantwortlich.

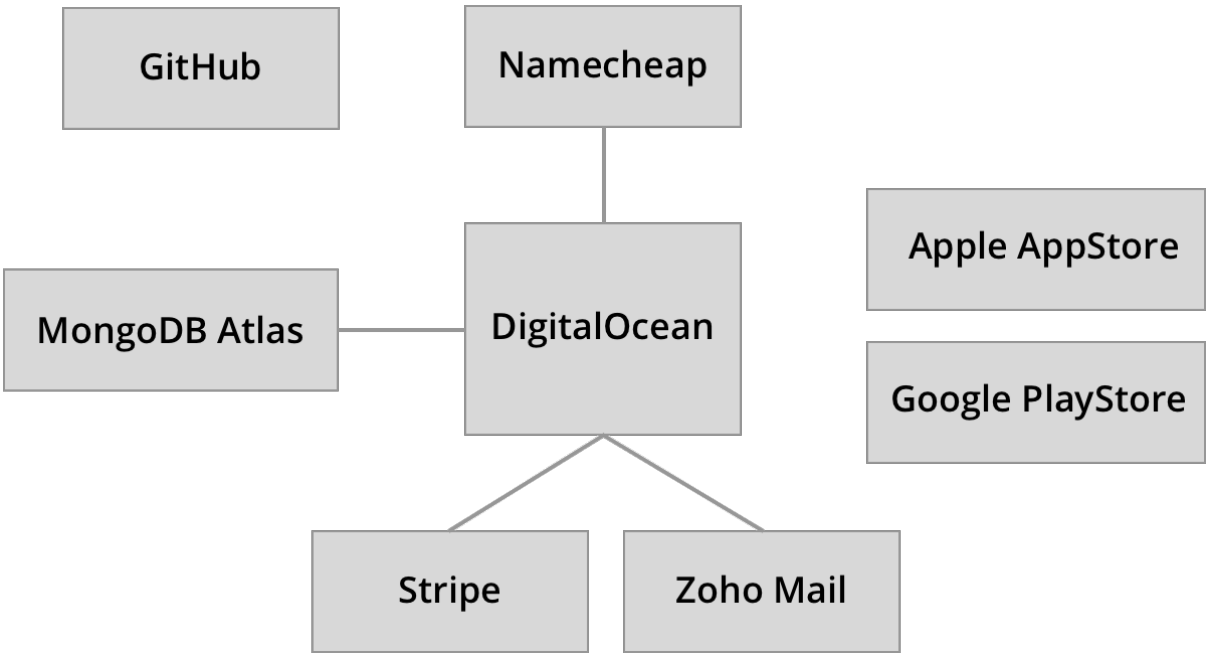


Abbildung 3: Services, die von Proludo verwendet werden

- **GitHub:** Als zentrales Code-Repository wird GitHub verwendet. Dort wird die gesamte Codebase und alle früheren Versionen davon gespeichert und verwaltet.
- **DigitalOcean:** Der Server ist in der Cloud, gehostet von dem Hosting Provider DigitalOcean. Er wird in Form eines vollwertigen Ubuntu-Containers, dem eine bestimmte Hardware zugewiesen ist, betrieben. Der Vorteil eines Cloud-Providers ist, dass die Server nicht gewartet werden müssen und dass diese sehr einfach skaliert

werden können. Sobald die Nutzerzahlen sich erhöhen und die momentan zugewiesene Hardware nicht mehr ausreicht, können z.B. mehr CPU-Kerne oder mehr RAM zugewiesen werden.

- **Namecheap:** Die Domain zusammen mit dem SSL-Zertifikat wurde bei Namecheap gekauft. Die Domain ermöglicht das Abrufen der Webseite über den Namen proludo.net. Das SSL-Zertifikat sorgt für eine sichere, verschlüsselte HTTPS-Verbindung und lässt im Browser neben der URL das Schloss erscheinen.
- **MongoDB Atlas:** Die Datenbank wird von dem Anbieter MongoDB Atlas auf AWS-Servern gehostet. Dieser Service übernimmt wöchentliche Backups und weitere Sicherheitsmassnahmen, damit die Datenbank ständig verfügbar, performant und sicher vor Datenverlust ist.
- **Zoho Mail:** Dieser Mail-Anbieter ist für das Senden und Empfangen von Mails von den Adressen info@proludo.net und support@proludo.net zuständig. Über die info-Adresse werden Mails zur Verifizierung und Erinnerungen zum Abo-Ende versandt. Wenn ein Nutzer ein Problem hat, kann er sich über die Support-Mail an mich wenden.
- **Stripe:** Um Zahlungen entgegenzunehmen, ist ein Zahlungsanbieter vonnöten. Stripe ist der weltweit bekannteste und bietet eine breite Menge an Funktionen. So lassen sich Abonnements für Nutzer erstellen und verwalten. Zahlungsinformationen und die Abo-Verwaltung laufen direkt über die Server von Stripe, auf die man von Proludo weitergeleitet wird.
- **Apple AppStore und Google PlayStore:** Um die Nutzer auf mobilen Geräten erreichen zu können, wurde die App in beiden Stores veröffentlicht.

3.5 Code

Um einen Einblick hinter die Kulissen zu ermöglichen, folgen nun einige Ausschnitte vom Proludo Source Code, der auf GitHub gespeichert ist. Ich habe drei Ausschnitte ausgewählt, die jeweils eine andere Funktion im Code haben, um eine möglichst grosse Bandbreite an Bausteinen der Applikation zu zeigen. Um dies verständlich darzustellen, habe ich relativ kurze und einfach verständliche Code-Ausschnitte gewählt. Die gesamte Codebase besteht aus ca. 25'000 Zeilen nur von mir geschriebenen Codes. Dazu kommen auch noch importierte Bibliotheken. Eine Bibliothek ist Code, der von anderen Personen geschrieben wurde und bestimmte Funktionen bereitstellt. Dieser Code kann von anderen Entwicklern importiert werden, um wertvolle Zeit zu sparen und die gewünschte Funktion mit wenigen Zeilen implementieren zu können.

Der folgende Ausschnitt gibt einen Einblick in die Struktur der Serverapplikation, die für die Verarbeitung der Daten zuständig ist. Diese API ist die zentrale Schnittstelle von Proludo, auf die von allen Geräten zugegriffen wird und von der Daten gesendet werden.

Listing 1: Agenda API

```
1 router.post("/api/agenda/create", auth, async (req, res) => {
2   try {
```

```
    const { date, task } = req.body;
    const agenda = await AgendaApp.findById(req.user._id);
    const index = agenda.data.findIndex((v) => v.date === date);
    if (index === -1) {
      agenda.data.push({ date, items: [task] });
    } else {
      agenda.data[index].items = [...agenda.data[index].items, task];
    }
    await agenda.save();
    res.status(200).send("Task created...");
  } catch (e) {
    res.status(400).end();
    console.error(e);
  }
});
```

Es handelt sich um einen Ausschnitt der Datei *server/routers/agenda.js*, in der alle API-Schnittstellen, die etwas mit der Agenda zu tun haben, untergebracht sind. Hier wird eine solche Schnittstelle definiert, nämlich unter der Adresse `https://proludo.net/api/agenda/create`. Diese hat den Typ POST-Request. Es gibt verschiedene Arten von HTTP-Requests, darunter der GET-Request, der verwendet wird, um Informationen von einem Server abzurufen, oder der POST-Request, um dem Server Informationen zu übergeben. Hier wird ein POST-Request verwendet, um dem Server die Daten eines neuen Agenda-Eintrages zu übergeben und in der Datenbank zu speichern. Mit der *AgendaApp.findById()*-Funktion wird nun auf die Datenbank zugegriffen, um die Agenda des Nutzers, von dem der Request stammt, zu finden. Mit *agenda.data.push()* wird nun der neue Eintrag der Agenda hinzugefügt und anschliessend mit *agenda.save()* in der Datenbank gespeichert. Zum Schluss wird der Statuscode 200 zurückgesendet, was soviel bedeutet wie ‘alles hat funktioniert‘. Wie man sieht, ist der ganze Codeblock in einem try-catch-Statement eingehüllt. Dieses fängt mögliche Fehler ein und sendet dann den Statuscode 400 zurück, was soviel bedeutet wie ”es ist etwas schief gegangen”.

Der nächste Ausschnitt ist auch Teil der Serverapplikation und soll einen Einblick in die Authentifizierung geben. Dabei wird überprüft, ob ein Nutzer dazu berechtigt ist, auf bestimmte Daten zuzugreifen oder Daten zu verändern.

Listing 2: Authentifizierung

```
const auth = async (req, res, next) => {
  try {
    const token = req.header("Authorization").replace("Bearer ", "");
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    const user = await User.findOne({
      _id: decoded._id,
      "tokens.token": token
    });
    if (!user) {
      throw new Error();
    }
    req.user = user;
    next();
  } catch (e) {
    res.status(401).send({ error: "Please authenticate" });
  }
};
```

Es handelt sich um einen Ausschnitt der Datei *server/middleware/paymentAuth.js*. Diese Funktion überprüft, ob der Nutzer berechtigt ist, einen bestimmten Inhalt abzurufen. Diesem Vorgang liegt das Prinzip des JSON Web Tokens zu Grunde. Vom Server wird eine verschlüsselte Zeichenkette generiert, die dann einem Nutzer zugeschickt wird. Die Daten des Nutzers können dann nur mit diesem Schlüssel abgerufen werden. In dieser Funktion wird dieser Schlüssel, der vom Nutzer gesendet worden ist, auf dem Server auf Echtheit überprüft. Dies geschieht mit der *jwt.verify()*-Funktion. Danach wird der Nutzer, dem dieser Token gehört, abgerufen. Falls bisher alles geklappt hat, kann der Nutzer die gewünschten Daten abrufen. Falls aber ein falscher oder gar kein Token gesendet wurde und der Nutzer somit nicht authentifiziert ist, wird dies im catch-Block abgefangen und ein Fehler mit dem Statuscode 400 wird zurückgeschickt.

Nun noch ein Ausschnitt der Webapp, um zu zeigen, wie die App selber, die der Nutzer sieht, aufgebaut ist.

Listing 3: Navigationsleiste

```
import ExitToAppIcon from "@material-ui/icons/ExitToApp";
import MenuIcon from "@material-ui/icons/Menu";
import React from "react";
import { Link, useHistory } from "react-router-dom";
import styled from "styled-components";
import LogoImg from "../../../assets/img/logo.svg";
import { MAIN_COLOR, MAIN_COLOR_DARK } from "../../../utils/theme";

export default function Topbar(props) {
  const history = useHistory();
  const { toggleSidebar } = props;
  return (
    <Header className="col-xs-12">
      <Link to="/" style={{ textDecoration: "none" }}>
        <Logo>
          <img src={LogoImg} alt="Logo" style={{ height: "14px" }} />
              proludo
        </Logo>
      </Link>
      <Spacer />
      <LogoutButton onClick={() => history.push("/logout")}>
        <ExitToAppIcon style={{ color: "#fff" }} />
      </LogoutButton>
      <MenuButton onClick={toggleSidebar}>
        <MenuIcon style={{ color: "#fff" }} />
      </MenuButton>
    </Header>
  );
}

const Header = styled.div...
const Logo = styled.div`
  color: #fff;
  font-size: 2em;
  font-weight: 600;
  padding: 5px;
  border-radius: 4px;
```

```
37    cursor: pointer;
38    &:hover {
39      background: ${MAIN_COLOR_DARK};
40    }
41  `;
42  const Spacer = styled.div...
43  const LogoutButton = styled.div...
44  const MenuButton = styled.div...
```

Diesen Code bekommt der Nutzer in Form eines Elements auf der Webseite zu Gesicht, deshalb enthält dieser auch verschiedene Style-Regeln, die bestimmen, wie eine Komponente genau aussieht. Dies ist ein Ausschnitt der Datei *src/components/layout/app/Topbar.js*, welche für die Darstellung und Logik der Navigationsleiste in der Web-App verantwortlich ist. Am Anfang der Datei werden verschiedene Bibliotheken und Objekte von anderen Dateien importiert, damit sie im Code verwendet werden können. Als nächstes werden verschiedene Style-Regeln definiert, die bestimmen, wie ein Element auf der Webseite aussieht. Dies ist gewöhnlicher CSS-Code. Was man als nächstes sieht, ist ein gewöhnlicher React Component: Ein Element auf der Webseite, das mit JavaScript und React erstellt wurde und wiederverwendbar ist. Es ist eine Funktion, die ein Objekt, das mit der JSX-Syntax geschrieben wurde, zurückgibt. Diese Syntax sieht exakt aus wie normale HTML-Syntax, ist aber in Wirklichkeit ein JavaScript-Objekt. Hier wird also die Navigationsleiste zurückgegeben, die das Proludo-Logo und verschiedene Buttons beinhaltet. Darunter z.B. der Button, um sich auszuloggen.

Nun gibt es auch noch den *mobile/*-Ordner, der allen Code für die Mobile Apps beinhaltet. Dieser ist mit React Native geschrieben, sieht allerdings dem React-Code vom dritten Beispiel sehr ähnlich. Der Unterschied ist, dass anstelle von HTML-Webkomponenten native Komponenten vom jeweiligen Betriebssystem, sprich Android oder iOS, verwendet werden. Diese werden von der Library ‘react-native‘ importiert. Ausserdem werden völlg andere Libraries verwendet. Diese sind in der Datei *package.json* alle aufgelistet.

3.6 Beta-Testing

Bevor die App veröffentlicht wurde, musste natürlich ordentlich auf Bugs getestet werden. Dazu gehören verschiedene Arten von Bugs. Zum Beispiel kann es sein, dass die App bei bestimmten Aktionen abstürzt, sich nicht so verhält wie sie sollte, oder Inhalt falsch darstellt. Viele dieser Fehler konnten vor der Veröffentlichung behoben werden, da mir 15 Leute geholfen haben, die App zu testen, und mich über Bugs informiert haben. Ich habe mehrere Beta-Versionen in den Stores und auf dem Web veröffentlicht, welche durchgetestet und dann wieder verbessert wurden, bis die App soweit war, dass mir keine weiteren schweren Bugs mehr bekannt waren und ich endlich die Version 1.0.0 veröffentlichen konnte.

3.7 Design

Beim Design wurde auf eine simple, benutzerfreundliche und moderne Oberfläche geachtet. Dabei wurden sowohl die Google Material Designgrundlagen als auch die Apple Human Interface Guidelines beachtet. Mir war sehr wichtig, eine ansprechende Benutzeroberfläche zu entwickeln, da dies ein elementarer Bestandteil des Nutzererlebnisses und somit des Erfolgs einer App ist.

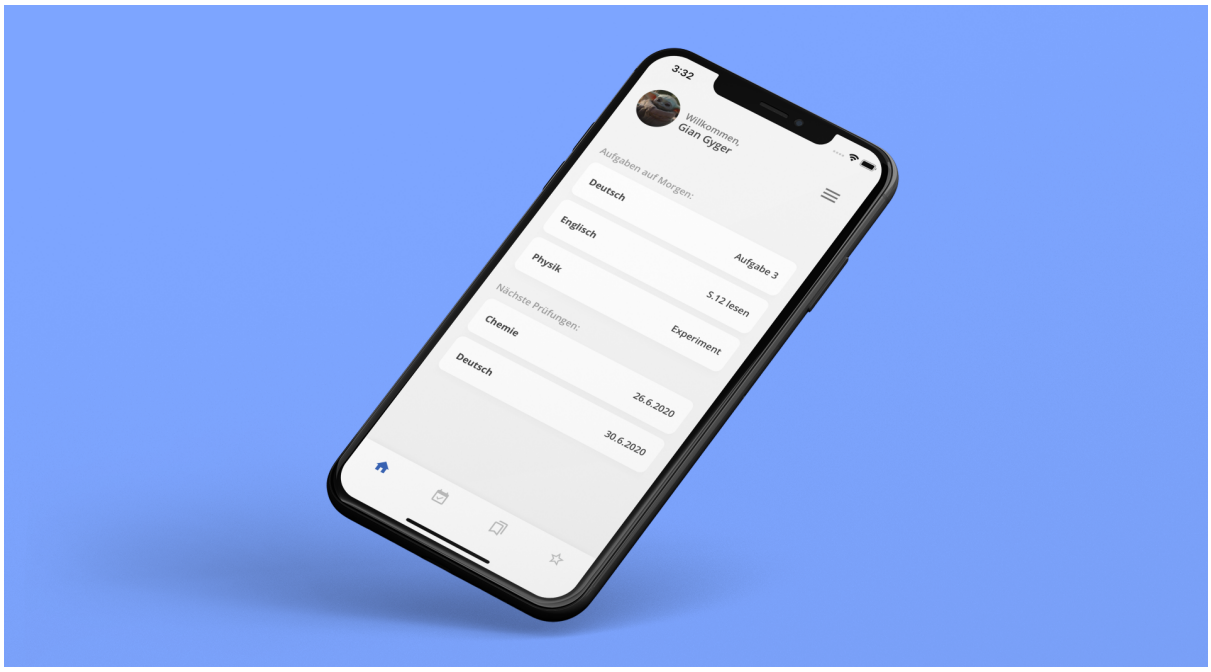


Abbildung 4: Design der Mobile App

3.8 Rechtliches

Beim Betreiben einer Webseite müssen verschiedene rechtlichen Anforderungen beachtet werden. Zum einen muss jede Webseite ein Impressum haben. Dies gibt Auskunft darüber, von wem die Seite betrieben wird und wie man die Person oder das Unternehmen erreichen kann. Zum anderen ist eine Datenschutzerklärung vonnöten, die darüber informiert, was genau mit persönlichen Daten passiert und ob diese gespeichert werden. Dies ist gerade für Proludo sehr wichtig, da Name, Email, Passwörter und weitere Nutzerinformationen wie Noten gespeichert werden. Um hier Sicherheit zu gewähren, werden natürlich alle Nutzerdaten verschlüsselt: sowohl in der Datenbank auf dem Server, als auch der lokalen Datenbank auf dem mobilen Gerät, und die Datenübermittlung durch das HTTPS-Protokol mit einem SSL-Zertifikat. Der zweite sehr wichtige Punkt sind die Allgemeinen Geschäftsbedingungen (AGB). Da Proludo ja ein bezahltes Abonnement beinhaltet, muss mit dem Nutzer ein Kaufvertrag abgeschlossen werden, welcher in den Geschäftsbedingungen geregelt ist. Dazu gehören aber auch Dinge wie der Haftungsausschluss. Es ist wichtig, dass ich nicht für Datenverlust, Falschinformationen oder andere Szenarien hafte, die eventuell auftreten könnten. Diese AGB habe ich mit Hilfe einer Vorlage eines anderen Unternehmens geschrieben und dann einem Anwalt zur Überprüfung geschickt.

3.9 Veröffentlichung

Nach dem Release der Version 1.0 und nachdem einige Leute die App heruntergeladen und einen Account erstellt haben, habe ich bemerkt, dass von Kunden zwar ohne Probleme ein Account erstellt wird, aber die meisten bei der Aufforderung, ihre Zahlungsinformationen anzugeben, aussteigen. Bis jetzt war es nämlich auch für die kostenlose Testversion nötig, die Zahlungsinformationen anzugeben, damit das Abo dann automatisch verlängert werden kann. Da dies die meisten Kunden jedoch nicht getan haben, habe ich mich dazu

entschlossen, die Möglichkeit einzubauen diesen Schritt zu überspringen. Die Angabe der Kreditkarte ist nun also erst bei der bezahlten Verlängerung des Abos nötig. Ich erhoffe mir dadurch, dass mehr Kunden das Produkt testen, dann davon überzeugt werden und dieses zuletzt in ihren Schulalltag einbauen. Danach ist es viel wahrscheinlicher, dass sie ihre Kreditkarte angeben und für das Produkt bezahlen.

3.10 Werbung

Um nun das fertig entwickelte Produkt auch unter die Leute zu bringen, sind Werbeaktionen erforderlich. Deshalb habe ich auf Instagram eine Werbekampagne gestartet, die eine Woche dauerte und ein Gesamtbudget von 150 Franken hatte. Damit hat die App ca. 150 neue Nutzer gewonnen und die Nutzerzahl erhöht sich dadurch bis heute täglich, obwohl keine Werbung mehr geschaltet wird. Dieser Effekt ist vermutlich darauf zurückzuführen, dass sich die App durch Empfehlungen in Klassen von selbst verbreitet. In den Sportferien 2021 ist eine grössere Werbekampagne in den sozialen Netzwerken und möglicherweise an anderen Schulen geplant, um Proludo eine grosse Reichweite und Anzahl Nutzer zu verleihen.

4 Produkt

4.1 Agenda

In der Agenda (siehe Abb. 5) können alle Aufgaben und Prüfungen eingetragen werden. Öffnet man diese in der App, bekommt man als erstes eine Übersicht über die aktuelle Woche. Hier kann man neue Aufgaben und Prüfungen hinzufügen. Ausserdem gibt es eine Gesamtübersicht, in der man alle anstehenden Aufgaben und Prüfungen gelistet sehen kann. Was die Agenda von anderen Apps unterscheidet, ist die Teilen-Funktion. Man kann nämlich seine eigene Agenda auf öffentlich einstellen oder auf die Agenda von anderen Nutzern zugreifen. Wenn man diese speichert, sieht man alle Einträge dieses Nutzers in seiner eigenen Agenda und kann diese als eine Art Klassenbuch verwenden. So muss nur noch eine Person die Aufgaben eintragen.

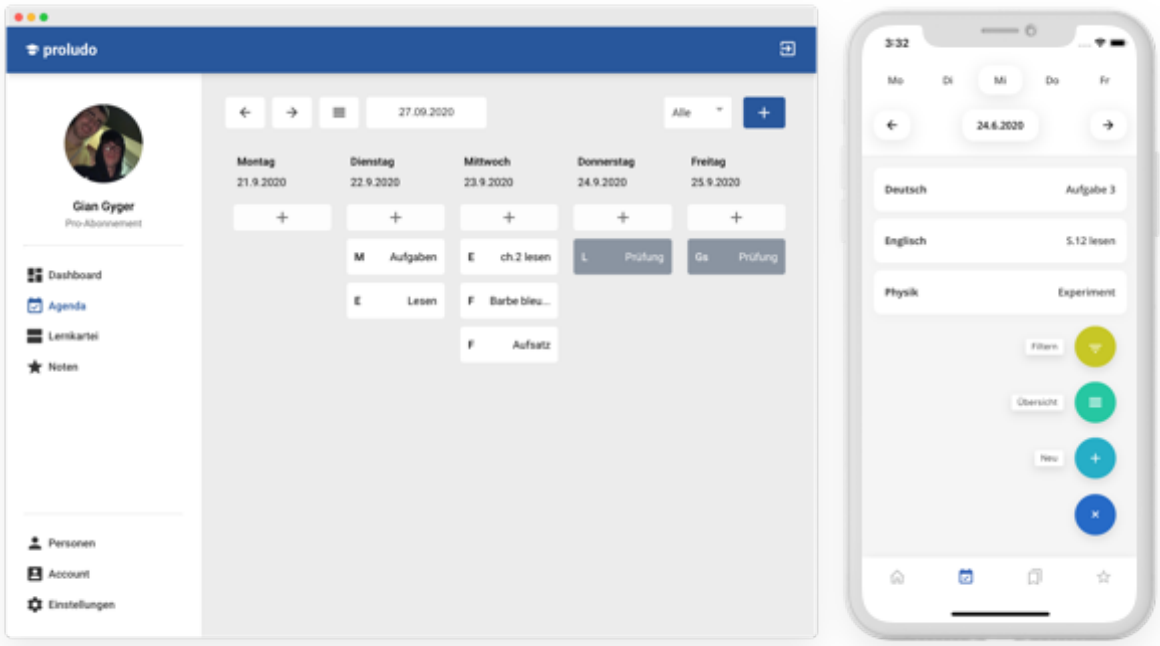


Abbildung 5: Agenda in Proludo

4.2 Lernkartei

In der Lernkartei (siehe Abb. 6) können Wörter mit diesen 4 verschiedenen Lernmethoden gelernt werden:

- **Kartei:** Hier gibt es vier Fächer. Alle Wörter befinden sich am Anfang im ersten Fach, wobei das Ziel ist, alle ins letzte Fach zu bringen. Wenn man ein Wort gewusst hat, kann man einfach nach rechts wischen, falls nicht nach links. Wischt man nach links, kommt das Wort wieder ins erste Fach.
- **Markieren:** Hier scrollt man durch alle Kärtchen und kann diese markieren, wenn man das Wort nicht weiss. Danach können alle markierten Kärtchen separat gelernt werden.

- **Schreiben:** Hier muss man das Wort korrekt schreiben. Nur wenn die Rechtschreibung stimmt, wird das Kärtchen als korrekt eingestuft.
- **Zuordnen:** In diesem Spiel wird ein Kärtchen angezeigt und man muss die korrekte Übersetzung von 4 angezeigten Lösungen auswählen.

Auch hier steht das Teilen im Vordergrund. So kann beim Erstellen eines Lernsets ausgewählt werden, ob die Kartei öffentlich sein soll oder nicht. Mit einer Suche kann man auf alle öffentlichen Lernkarteien der gesamten Community zugreifen und diese lernen. Auch bietet die App eine Import-Funktion von Quizlet und CSV-Dateien. So können bestehende Lernsets einfach in Proludo importiert werden.

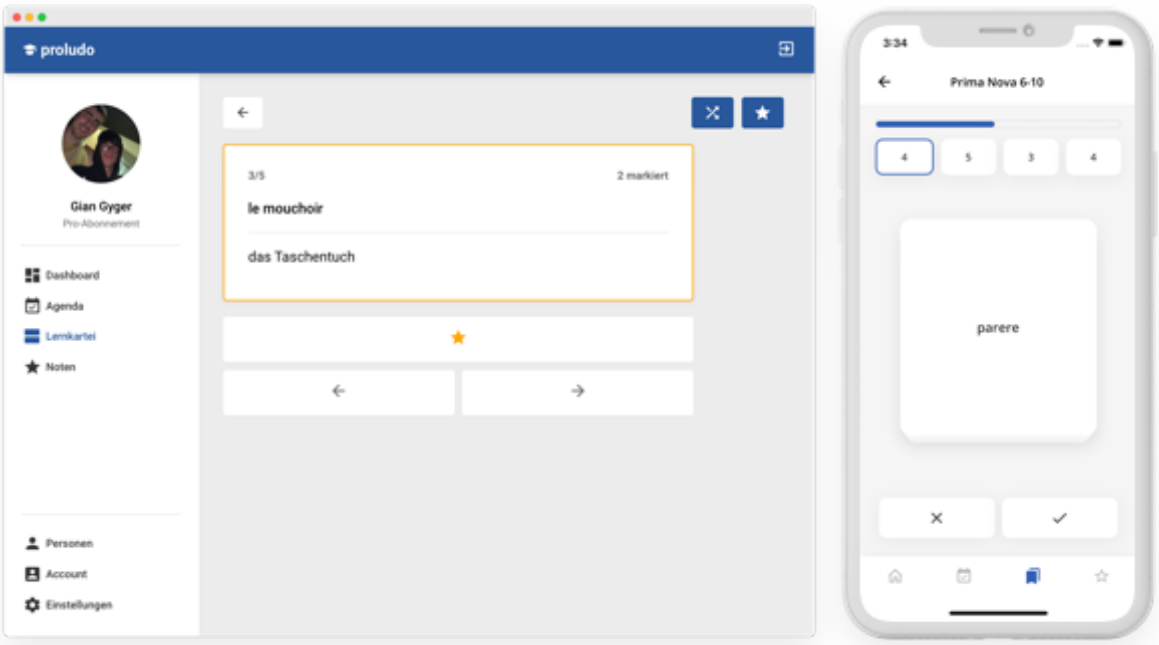


Abbildung 6: Lernkartei in Proludo

4.3 Notenverwaltung

In der Notenverwaltung (siehe Abb. 7) können alle Fächer und Noten eingetragen werden. Es werden dann alle Notenschnitte und – falls man in den Einstellungen das Schweizer Notensystem ausgewählt hat – auch Pluspunkte berechnet. Ausserdem kann man mit dem Notenrechner ausrechnen, welche Noten man für einen bestimmten Schnitt benötigt. Zusätzlich gibt es einige Diagramme, die die Leistungen graphisch darstellen.

5 Rückblick

Gestartet habe ich mit dem Projekt bereits im Dezember 2019, unabhängig von meiner Maturaarbeit. Da ich bereits etwas Erfahrung mit React und Webentwicklung an sich hatte, war der Einstieg leicht und ich konnte schnell einen ersten Prototypen entwickeln. Im März konnte ich das Projekt dann zu meiner Maturaarbeit machen. An diesem Punkt war ein erster Prototyp der Webapp mit den grundlegenden Funktionen und einem

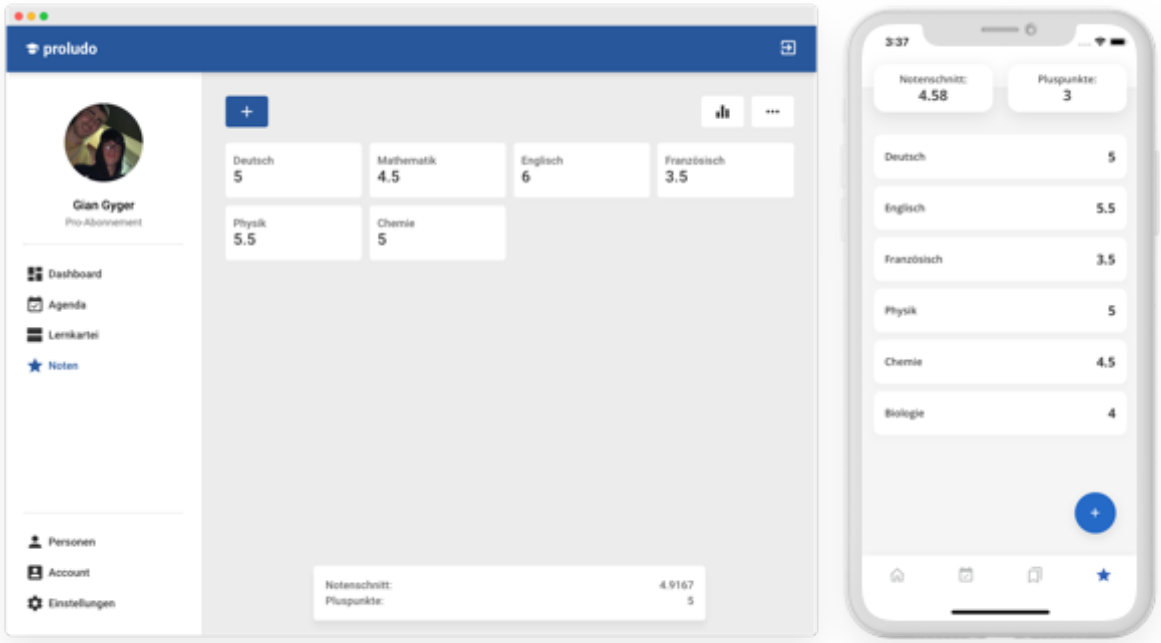


Abbildung 7: Notenverwaltung in Proludo

Back-End bereits entstanden. Ende April waren die meisten Features in der Web-App fertiggestellt und ich machte mich an die Performance-Optimierung des Back-Ends. Im Mai habe ich dann mit der Entwicklung der Mobile App begonnen. Ursprünglich dachte ich, dafür würde ich nur etwa 2 Monate benötigen, das Ganze hat sich aber stark in die Länge gezogen, da ich gemerkt habe, dass die ganze App sehr komplex geworden ist und viele Features noch nicht implementiert waren. Dazu kamen viele Dinge, die ich am Anfang nicht eingeplant hatte, wie alles rund um die Entwicklung selbst, wie Datenschutzerklärung und AGB. Vor allem aber hat mir das Problem des Offline-Supports in der Mobile App einen Strich durch die Rechnung gemacht. Um dies vollständig zu lösen, habe ich fast 2 Monate gebraucht. Ausserdem hatte die Mobile App wie auch die Web App noch viel zu viele Bugs, um diese veröffenlichen zu können. Deshalb habe ich mehrere Tests mit ca. 15 Nutzern durchgeführt, was auch über einen Monat gedauert hat. Veröffentlichen wollte ich das Projekt ursprünglich Ende Juni, das war aber, wie ich schnell realisiert habe, nicht im Bereich des Möglichen und hat sich schlussendlich bis Mitte Oktober hinausgezögert. Eine weitere Schwierigkeit war die Implementierung des Zahlungssystems, was wegen der komplexen API und den vielen Möglichkeiten relativ lange gedauert hat. Nachdem auch nach dem Release noch einige Bugs behoben waren, war ich endlich mit dem Produkt zufrieden und konnte mit der Werbung auf Instagram und Google beginnen. Die Entwicklung hat also fast ein ganzes Jahr gedauert.

Ich habe dabei viele neue Technologien kennengelernt und vor allem gelernt, wie eine solch komplexe produktive Applikation strukturiert und verwaltet werden muss. Ich habe gelernt, wie wichtig es ist, übersichtlichen, gut strukturierten und kommentierten Code zu schreiben, um eine grosse Codebase mit ca. 25000 Zeilen Code zu verwalten. Im Rückblick hätte ich von Anfang an viel mehr Wert darauf legen sollen.

6 Ergebnisse und Zukunft

Bis zum heutigen Stand (5.12.2020) sind über 200 Nutzer registriert. Nun gilt es herauszufinden, ob und wie schnell die Werbeausgaben wieder zurückfliessen und ob es sich überhaupt lohnt, auf sozialen Netzwerken Werbung zu schalten.

Nun ist das Ziel, die App weiter zu verbessern und die Nutzeranzahl auf etwa 1000 aktive und abonnierte Nutzer zu erhöhen. Danach können beliebige weitere Features implementiert werden. Dies sind Features, mit denen ich die Plattform erweitern könnte:

- Notizen, an denen gemeinsam gearbeitet werden kann;
- ein Stundenplan, der die nächste Schulstunde und die Zimmernummer live anzeigt und auch den Überblick über die Woche bietet;
- Zusammenfassungen zu verschiedensten Themen erstellen, die von Nutzern geteilt werden können;
- ein Abonnement für Lehrpersonen, das es ermöglicht Klassen zu verwalten - so können Lernsets mit der Klasse geteilt und die App als Klassenbuch benutzt werden

Ich könnte mir vorstellen, das Produkt in der Zukunft auch an Schulen zu verkaufen. Wer weiss, vielleicht ist Proludo ja bald ein Standardtool, das von Schülern und Lehrpersonen benutzt wird – nicht nur in der Schweiz, sondern weltweit. Die Möglichkeiten sind endlos...

A Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während meiner Maturitätsarbeit unterstützt haben. Mein Dank geht speziell an die folgenden Personen:

Michael Liebich, meinem Informatiklehrer und Betreuer dieser Arbeit, der mich während des Prozesses unterstützt hat.
Philip Steinemann, meinem Klassenkameraden, der mich beim Marketing unterstützt hat.

Die 15 Schüler und Schülerinnen, die sich für das Testen bereitgestellt haben und mir wertvollen Input zur Verbesserung der App gegeben haben.

B Anhang

1. Webseite: `https://proludo.net`
2. iOS App: `https://apps.apple.com/ch/app/proludo/id1519972105`
3. Android App: `https://play.google.com/store/apps/details?id=net.proludo.student`
4. GitHub Repository: `https://github.com/Gian-Gyger-Software/Proludo`
5. Proludo App Trailer: `https://youtu.be/H2j3eRbbgHk`
6. Proludo Tutorial: `https://youtu.be/IQ8LBDhlO6U`

C Quellen

1. Stack Overflow: `https://stackoverflow.com/`
2. MDN: `https://developer.mozilla.org/en-US/docs/`
3. React Docs: `https://reactjs.org/docs/getting-started.html`
4. React Native Docs: `https://reactnative.dev/docs/getting-started`
5. Express Docs: `https://expressjs.com/de/api.html`
6. MongoDB Docs: `https://docs.mongodb.com/`
7. Stripe Docs: `https://stripe.com/docs`

D Eigenständigkeitserklärung

Ich, Gian Gyger, erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und ohne Benützung anderer als der angegebenen Quellen und Hilfsmittel verfasst beziehungsweise gestaltet habe.

.....