Projektdokumentation

31. Juli 2023

Julius Beutel (jb266), Maik Bucher (mb389),
Maurice Dolibois (md147) und Samuel Riester (sr185)

Anwendungsprojekt unter der Betreuung von Korbinian Kuhn

Inhaltsverzeichnis

Idee und Motivation

Unsere Idee für das Projekt "Bazarify" entstand aus der Notwendigkeit, das jährliche Basar-Event von Maiks Skiverein (Skiverein Laichingen) zu modernisieren und zu optimieren. Der aktuelle Ablauf des Basars, bei dem Produkte von Teilnehmern angenommen und weiterverkauft werden, geschieht analog und erfordert viel manuelle Arbeit. Unser Ziel ist es, eine benutzerfreundliche Software zu entwickeln, die den gesamten Basarprozess verbessert und effizienter gestaltet.

Die Software soll auf einem Laptop laufen und es soll möglich sein, den Basar in Echtzeit zu verwalten, Produkte zu erfassen, Bestände zu aktualisieren und Verkäufe zu registrieren, ohne dabei auf Papierunterlagen angewiesen zu sein.

Das bisherige analoge System der Skischule Laichingen ist veraltet, komplex und ineffizient. Mit unserer Software wird das Verwalten des Basars für alle Beteiligten erheblich einfacher und ressourcenschonender. Durch die digitale Erfassung und Verarbeitung der Informationen werden Fehler minimiert und die gesamte Organisation verbessert.

Da wir das Projekt im Rahmen eines Semesterprojekts entwickeln, entstehen für Maiks Skiverein keine Kosten. Die Hauptergebnisse des Projekts bestehen daher aus dem Erwerb von neuem Wissen und der Bereitstellung einer maßgeschneiderten Softwarelösung für die Skischule Laichingen. Unser Ziel ist es, den Basarprozess zu optimieren und eine zukunftsfähige Lösung zu schaffen, die den jährlichen Basar zu einem reibungslosen und erfolgreichen Ereignis macht.

Technischer Hintergrund

Frontend

- React wurde gewählt, weil es eine leistungsstarke und flexible JavaScript-Bibliothek ist, die eine reaktive Benutzeroberfläche ermöglicht. Mit React können wir komponentenbasierte Ansätze verwenden, was die Code-Wartung und Organisation erleichtert.
- Für die Infografik haben wir uns für Recharts entschieden, da es eine benutzerfreundliche und anpassbare Bibliothek für die Erstellung von Diagrammen und Grafiken ist. Die umfangreiche Dokumentation und die Vielzahl von Diagrammtypen, die Recharts bietet, ermöglichen uns eine ansprechende und informative Visualisierung unserer Daten.
- TailwindCSS wurde gewählt, weil es ein modernes Utility-First CSS-Framework ist, das es uns ermöglicht, schnell und effizient ein konsistentes und ansprechendes Design zu erstellen. Die Flexibilität und das responsive Design von TailwindCSS erleichtern die Anpassung der Benutzeroberfläche an verschiedene Bildschirmgrößen.

Backend

- Next.js wurde als Backend-Framework gewählt, da es auf React basiert und eine serverseitige Rendering-Funktionalität bietet. Dies ermöglicht eine schnellere Ladezeit und eine bessere Suchmaschinenoptimierung (SEO) für unsere Webanwendung.
- Node.js wurde aufgrund seiner Geschwindigkeit und Skalierbarkeit als Plattform für das Backend ausgewählt. Da Node.js eine ereignisgesteuerte, nicht blockierende Architektur hat, können wir effiziente und schnelle Anfragen an den Server verarbeiten.
- Express.js wurde als Webframework gewählt, weil es leichtgewichtig ist und eine einfache Möglichkeit bietet, Routen und Middleware zu definieren. Express.js ermöglicht uns die schnelle Entwicklung von APIs und das effiziente Verwalten von HTTP-Anfragen.
- MongoDB wurde als Datenbank gewählt, da es eine dokumentenorientierte, NoSQL-Datenbank ist, die eine flexible Strukturierung der Daten erlaubt. Die

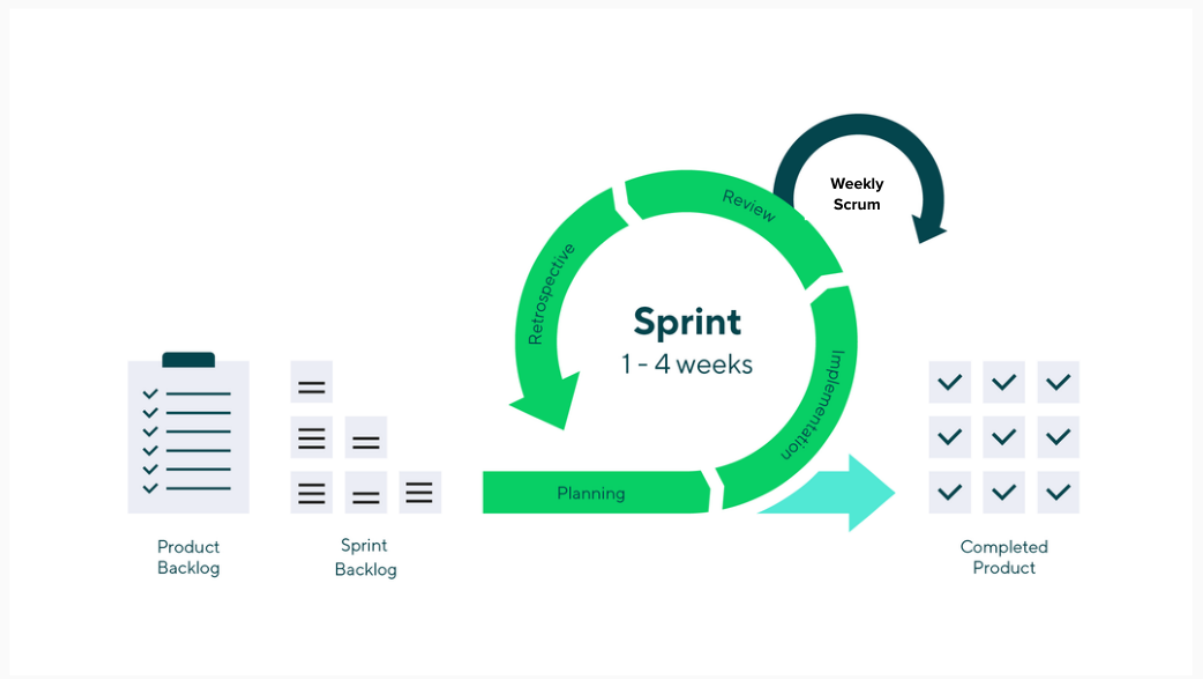
skalierbare Natur von MongoDB ermöglicht es uns, mit dem Wachstum der Anwendung umzugehen und eine effiziente Datenverwaltung zu gewährleisten.

- Mongoose wurde als ODM (Object Document Mapper) für MongoDB verwendet, da es die Interaktion mit der Datenbank vereinfacht und eine klare, schemaähnliche Struktur für unsere Datenmodelle bietet. Mongoose erleichtert die Validierung, Abfrage und Verwaltung von Daten und trägt zu einer stabilen Datenbankstruktur bei.

Die Kombination dieser Technologien ermöglicht es uns, eine leistungsfähige und benutzerfreundliche Web-App zu entwickeln, die sowohl auf der Client- als auch auf der Serverseite effizient arbeitet und eine ansprechende Benutzeroberfläche mit informativen Infografiken bietet.

Projektmanagement

Bei unserem Anwendungsprojekt haben wir für agiles Projektmanagement das Scrum-Prinzip verwendet. Dafür haben wir Jira und Confluence genutzt. Da wir uns nicht jeden Tag getroffen haben, haben wir uns entschieden, kein Daily Scrum zu machen, sondern einen Weekly Scrum. Das Weekly Scrum bestand aus Updates von jedem Teammitglied und auch außerordentlichen Themen.



Rollen

- Project Owner: Der Project Owner vertritt die Interessen des Kunden gegenüber des Scrum Masters und des Teams.
- Scrum Master: Ist zuständig für die Developer und für die Planung und Durchführung von den Meetings. Ebenso besorgt der Scrum Master benötigte Hardware oder Software.
- Developer: Sind das eigentliche Team. Die Developer planen mit dem Scrum Master die Aufgaben in einem Sprint und machen einen Weekly Scrum, um Zwischenstände auszutauschen.

Rollenverteilung

Die Rollenverteilung innerhalb des Teams wurde so realisiert, dass Samuel und Julius hauptsächlich das Frontend und Design übernommen haben und Maurice und Maik das Backend. Natürlich gab es breitere Aufgaben, die dazu geführt haben, dass alle Mitglieder auch mal in jedem Bereich gearbeitet haben, sodass jeder einen guten Überblick über das komplette System hat. Des Weiteren hat Maik zusätzlich die Rolle des Scrum Masters übernommen, um das Umsetzen des agilen Projektmanagements zu verwirklichen. Die Rolle der Developer haben alle Teammitglieder erfüllt.

- **Kernaufgaben von Julius**: Entwicklung der Frontend-Logik und Kommunikation mit dem Backend
- **Kernaufgaben von Maik**: Projektleitung und Organisation, Einbindung von Docker, sowie Entwicklung des Backends und der Datenbank
- **Kernaufgaben von Maurice**: Entwicklung des Backends und der Datenbank, sowie die Kommunikation zwischen Frontend und Backend
- **Kernaufgaben von Samuel**: Entwicklung des UI-Designs samt Prototypen, sowie Entwicklung der Frontend-Komponenten und Frontend-Logik

Events & Backlogs

Project Backlog

Im Project Backlog werden die Issues erstellt und gelagert. Je nachdem können Issues bearbeitet, neu hinzugefügt oder gelöscht werden. Jeder aus dem Team, also auch die Developer, können Issues erstellen und bearbeiten.

Sprint Planning

Im Sprint Planning planen die Developer mit dem Scrum Master zusammen, was in dem kommenden Sprint alles an Issues abgearbeitet werden soll. Dies wird im Sprint Backlog festgehalten. Dabei werden auch alte Issues die nicht fertig gestellt werden konnten (sogenannte Spill-Overs) mit übernommen. Ein Sprint Planning passiert immer, wenn der letzte Sprint zu Ende ist und das Sprint Review und Sprint Retrospektive durchgeführt wurde.

Sprint Backlog

In diesem Dokument werden alle Issues aufgeschrieben, die in dem aktuellen Sprint abgearbeitet werden sollen. Nur Developer und Scrum Master haben auf das Sprint Backlog Zugriff.

Weekly Scrum

Im Weekly Scrum werden Zwischenstände, Probleme, Fragen und Weiteres besprochen. Dies wird in einem Dokument festgehalten und zur Dokumentation abgelegt.

Sprint Review

Im Sprint Review werden die Issues vom Sprint Backlog besprochen. Es wird geschaut, ob die Issue ordnungsgemäß erledigt worden sind oder ob etwas noch nicht ganz fertig geworden ist oder der Product Owner nicht zufrieden ist. Es können hier schon Spill-Overs definiert werden, die dann in den nächsten Sprint übernommen werden.

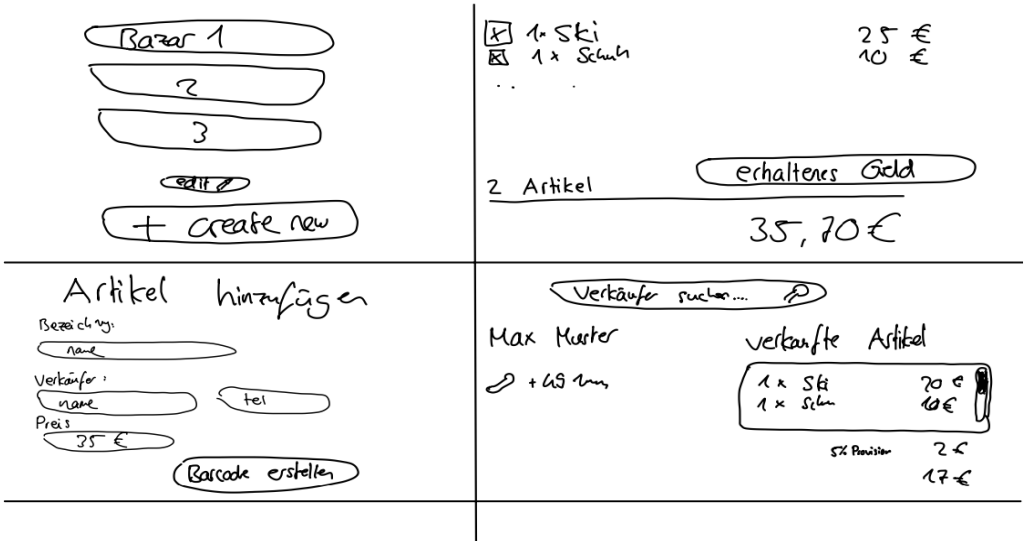
Sprint Retrospektive

In der Retrospektive werden die Prozesse des letzten Sprints analysiert. Mit der "Start-Stop-Continue"-Methode. Dies führt zu einer besseren und effizienteren Vorgehensweise im nächsten Sprint.

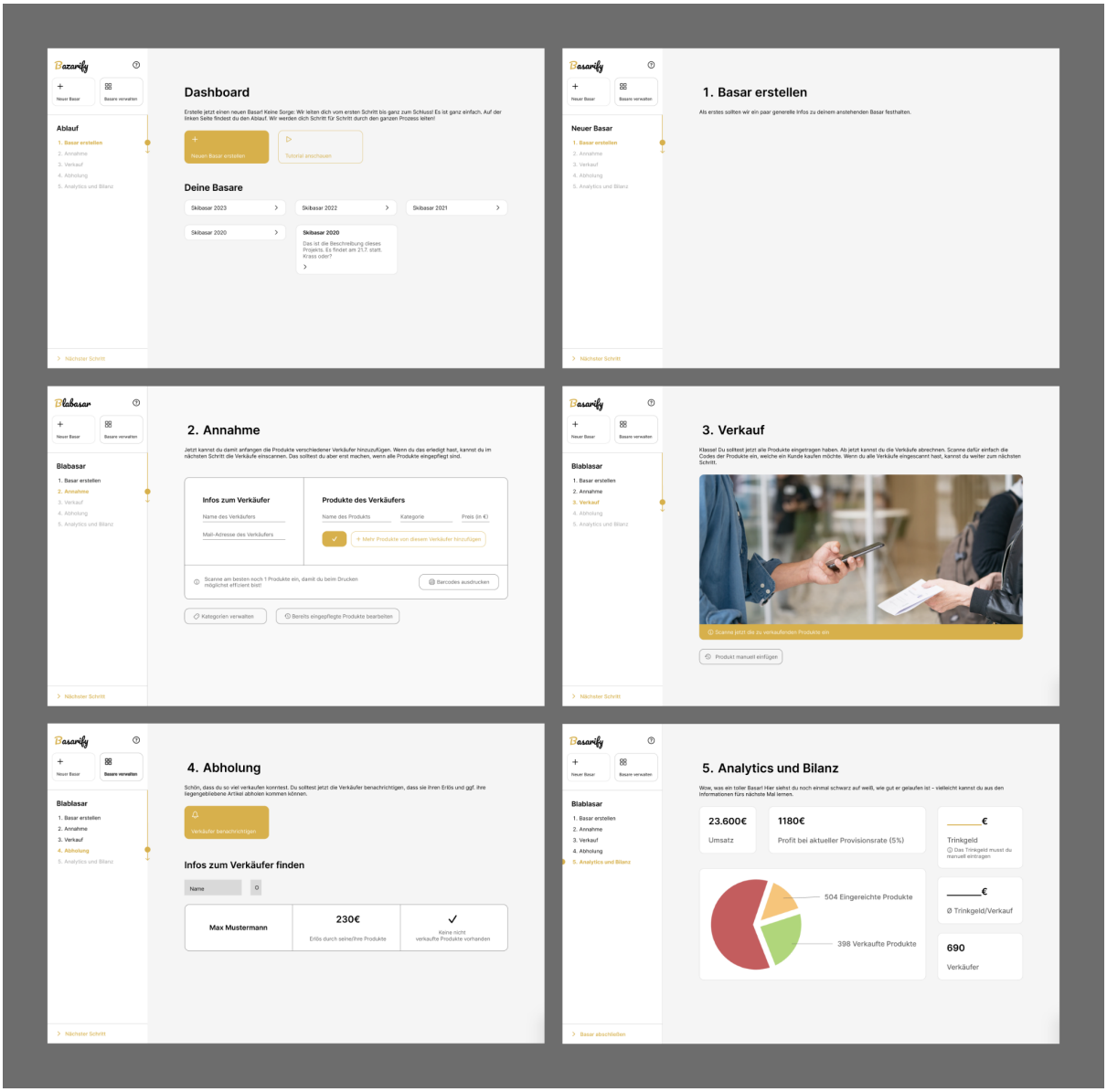
Verlauf des Projekts über die Zeit

Unser Projekt durchlief verschiedene Phasen, um eine optimale Lösung für die Skischule Laichingen zu entwickeln:

1. Anforderungssammlung: Wir begannen mit einem Lastenheft und engagierten uns in intensiven Besprechungen mit der Skischule Laichingen, um ihre Anforderungen und Bedürfnisse zu verstehen.
2. Designmethode Crazy 8s: Um möglichst viele kreative Lo-Fi-Designs zu sammeln und die besten Ideen zu erkunden, setzten wir die Crazy 8s-Methode ein.



3. Entwicklung eines Hi-Fi-Prototypen: Mithilfe von Figma erstellten wir einen ersten Hi-Fi-Prototypen mit einigen bereits ziemlich detaillierten Screens, um die Benutzeroberfläche zu visualisieren. Im Verlauf des Projekts kam es sogar zur Entwicklung einer zweiten Iteration des Designs, welches Probleme der ersten Iteration ausbesserte und weitere Verbesserungen mit sich brachte. Manche Designkomponenten wurden im Prototyp absichtlich ausgelassen, da wir für diese speziellen Komponenten entweder Vorlagen hatten oder erst schauen mussten, inwiefern wir sie tatsächlich möglichst effektiv bauen können.



Hier sind einige Screenshots dieses Prototyps. Unter diesem Link („https://www.figma.com/file/cm4z2naSfGHVYIzF45l6r9/Prototype?type=design&node-id=79%3A35&mode=design&t=0JAQmJpxwtruRqfW-1“) kann der komplette Prototyp (samt beider Versionen) begutachtet werden. Die finale Version unserer App sieht im Endeffekt ein kleines bisschen anders aus, diese kann beim Nutzen der App angeschaut werden.

4. Projektbeginn: Das Frontend-Team startete die Entwicklung, indem es häufig verwendete Komponenten für die vorhandenen Seiten erstellte und diese stilisierte. Gleichzeitig setzte das Backend-Team die Datenbank auf und schrieb die Anbindung.

5. Stetiger Fortschritt: Im Laufe der Zeit wurden immer mehr Komponenten entwickelt und Backend-Funktionalitäten implementiert. Wir integrierten auch den Barcode-Scanner, um das Benutzererlebnis zu verbessern.
6. Veranstaltungen: Am Präsentationstag bzw. der Medianight präsentierten wir unser bereits vorzeigbares und grob funktionierendes Produkt, das bereits viele Anforderungen erfüllte.
7. Endphase: In der finalen Phase des Projekts machten wir den Code cleaner und optimierten ihn. Außerdem arbeiteten wir an den letzten Feinheiten.
8. Abschluss: Schließlich finalisierten wir das Projekt, schrieben die letzten Tests und eine umfassende Dokumentation, um sicherzustellen, dass das Endprodukt gut dokumentiert und einsatzbereit ist.

Der Verlauf des Projekts spiegelt die Arbeit und den kontinuierlichen Fortschritt unseres Teams wider. Durch enge Zusammenarbeit und eine effiziente Entwicklungsstrategie konnten wir "Bazarify" erfolgreich vom Anfangsstadium bis zum voll funktionsfähigen und präsentationsreifen Produkt entwickeln.

Probleme und Challenges

Unser Projekt brachte verschiedene Herausforderungen mit sich, die wir gemeinsam erfolgreich bewältigt haben. Einige dieser Probleme waren:

1. Zeitmanagement und Zeitdruck: Obwohl es überraschenderweise nie ein schwerwiegendes Problem war, erforderte das Projekt dennoch ein gutes Zeitmanagement und eine effiziente Planung, um die gesteckten Ziele zu erreichen. Das motivierte und gut funktionierende Team trug dazu bei, dass wir den Zeitrahmen erfolgreich einhalten konnten.
2. Einbindung von Docker: Die Implementierung von Docker stellte eine Herausforderung dar, da es einige komplexe Konfigurationen und Abstimmungen erforderte, um die Containerisierung reibungslos in unsere Anwendung zu integrieren. Dennoch ermöglichte Docker letztendlich eine verbesserte Skalierbarkeit und Portabilität.
3. Clean Code: Die Idee, Clean Code am Ende des Projekts zu implementieren, erwies sich im Nachhinein als herausfordernd und weniger sinnvoll. Die kontinuierliche Wartung und Optimierung des Codes während der Entwicklung wäre möglicherweise effizienter gewesen.
4. Druckerprobleme: Ein unerwartetes Problem war, dass der Drucker keine einzelnen Barcodes drucken konnte, sondern diese nur gesammelt auf einer DIN A4 Seite ausgab. Dies erforderte eine zusätzliche Anpassung, um die Etiketten effizient zu drucken und Ressourcen zu sparen.
5. Merge-Konflikte: Während des Mergens traten gelegentlich Konflikte auf dem Develop-Branch auf, besonders wenn nicht zuvor die neueste Version von Develop auf den entsprechenden Branch gemergt wurde. Dies erforderte eine sorgfältige Abstimmung und Kommunikation innerhalb des Teams, um Konflikte zu lösen und einen reibungslosen Entwicklungsfluss sicherzustellen.
6. Backend Tests: Da die Datenbank lokal läuft, müssen auch die Tests, die mit der Datenbank zusammenhängen lokal ausgeführt werden, was bedeutet, dass sie nicht in einer automatisierten CI/CD-Pipeline, wie zum Beispiel in GitLab Actions, integriert werden können. GitLab läuft auf einem Server und kann dementsprechend keine Verbindung zum Localhost aufbauen. Infolgedessen wird der Test für Datenbankoperationen nicht in die GitLab CI/CD Pipeline aufgenommen, da er dort nicht erfolgreich durchgeführt werden kann. Dies haben wir durch eine Veränderung

im Pipeline-Skript geschafft, indem die betroffenen Tests durch RegEx herausgefiltert werden. Dennoch kann man lokal durch “npm test” im Backend Ordner die API-Schnittstellen testen

Trotz dieser Herausforderungen und Probleme haben wir als Team erfolgreich zusammengearbeitet, kreative Lösungen gefunden und das Projekt "Bazarify" erfolgreich abgeschlossen. Die Überwindung dieser Hindernisse hat uns wertvolle Erfahrungen und Erkenntnisse gebracht, die unsere Fähigkeiten und unser Wissen erweitert haben.

Lessons Learned

- Die Erfahrung, ein Projekt von Anfang bis Ende durchzuführen und die Erwartungen eines "Kunden" zu erfüllen, war äußerst lehrreich.
- Die klare Zuweisung von Rollen und Verantwortlichkeiten funktionierte gut und half bei einer effizienten Aufgabenbewältigung.
- Regelmäßige wöchentliche Meetings und Sprint Reviews waren äußerst hilfreich, um motiviert und fokussiert zu bleiben. Insgesamt war das Zeitmanagement und die Projektorganisation gut, obwohl das häufige Wechseln zwischen Notion und Jira bzw. Confluence verbessert werden könnte.
- Obwohl es zeitlich schwierig war, hätten wir noch mehr Gelegenheiten nutzen können, um gemeinsam vor Ort zu programmieren, da dies die Teamdynamik und den Fortschritt verbessert hätte.
- Das praktische Anwenden von Theoriewissen, wie das Erstellen eines Prototyps mit mehreren Iterationen, half dabei, eine immer bessere Lösung zu entwickeln.
- Das Programmier-Wochenende erwies sich als äußerst hilfreich für die Team-Einheit, den Fortschritt und die Motivation.
- Man sollte Merge-Konflikte vermeiden, indem man zuerst pullt und dann pusht. Generell sollte man noch mehr auf separaten Branches für separate Zwecke arbeiten.
- Die Zusammenarbeit mit einem Team aus Freunden machte das Projekt insgesamt noch angenehmer und erfolgreicher.
- Die Bedeutung von Clean Code wurde deutlich, und es wäre ratsam, damit von Anfang an zu arbeiten und nicht erst kurz vor der Abgabe.

Ausblick

Wir sind gespannt darauf, die Software beim nächsten Skibasar des Skivereins Laichingen einzusetzen und freuen uns auf das Feedback der Skivereinsmitglieder und Teilnehmer.

Eine sinnvolle Verbesserung für die Zukunft wäre die vollständige Responsivität der Software und möglicherweise eine spezielle Version, die sich auf Smartphones optimiert. Eine solche Version würde neue Use Cases eröffnen und den Nutzern ermöglichen, Bazarify bequem per Smartphone zu nutzen, indem sie QR-Codes mithilfe der Handy-Kamera scannen. Dieser Schritt würde die Benutzerfreundlichkeit erhöhen und das Produkt zukunftssicher machen. Außerdem wäre es hilfreich, wenn die App über mehrere Computer (Laptops) innerhalb desselben Netzwerks kommunizieren könnte, wodurch man mehrere “Kassen” bei einem Basar nutzen könnte.

Obwohl wir derzeit keine weiteren Versionen planen, da das Projekt im Rahmen des Anwendungsprojekts entstanden ist und unsere Teammitglieder in naher Zukunft wahrscheinlich nicht genug Zeit für die Weiterentwicklung haben, werden wir das Feedback und die Erfahrungen nutzen, um das Projekt "Bazarify" kontinuierlich zu verbessern. Wir möchten sicherstellen, dass die Software reibungslos funktioniert und den Bedürfnissen der Nutzer entspricht, auch wenn keine größeren Updates geplant sind.

Unser Ziel ist es, dass "Bazarify" erfolgreich beim nächsten Skibasar genutzt wird und die Skischule Laichingen von der verbesserten Organisation und Effizienz profitiert. Wir sind dankbar für die Möglichkeit, dieses Projekt durchzuführen, und werden die Software entsprechend den Ressourcen und Gegebenheiten weiterhin unterstützen, um eine erfolgreiche und nützliche Lösung zu bieten.