



Duale Hochschule Baden-Württemberg
Mannheim

Seminararbeit

Xpire - Dokumentation zum Projekthergang

Studiengang Wirtschaftsinformatik

Studienrichtung Software Engineering

Verfasser:
Engel, Andrea: 1757640
Dutzi, Jonas: 6681598
Richarz, Verena: 7983539
Waage, Felix: 3459154
Werner, Yvonne: 8519757
Westphal, Fabio: 6198411
Zahn, Milena: 7488221

Firma: SAP SE
Kurs: WWI 17 SEB
Studiengangsleiter: Prof. Dr. Sebastian Ritterbusch
Modul: Entwicklung mobiler Applikationen
Dozent/-in: Michael Spengler
Bearbeitungszeitraum: 11.05.2020 - 31.07.2020

Inhaltsverzeichnis

Abkürzungsverzeichnis	ii
1 Einleitung	1
1.1 Grundidee von Xpire	1
1.2 Was ist eine PWA?	2
1.3 Vorgehensweise	3
2 Analyse	4
2.1 Value Proposition Canvas	4
2.2 Benutzerprofilanalyse	6
2.3 Anforderungsanalyse	7
2.3.1 Funktionale Anforderungen	7
2.3.2 Nicht-funktionale Anforderungen	7
2.4 Monetarisierungsstrategie	8
3 Konzeption und Entwurf	10
3.1 Eingesetzte Technologien	10
3.1.1 React	10
3.1.2 Datenbank	10
3.1.3 Platform	11
3.1.4 Sonstiges	12
3.2 Konzeptuelles Modell	13
3.2.1 Notifications	14
3.3 Entwicklung von Prototypen	15
3.3.1 Figma	15
3.3.2 Mobile First	15
4 Präsentation der PWA	18
4.1 Anforderungsabgleich	23
4.1.1 Abgleich der funktionalen Anforderungen	23
4.1.2 Abgleich der nicht-funktionalen Anforderungen	24
5 Zusammenfassung und Ausblick	25
Literatur	28

Abkürzungsverzeichnis

DHBW Duale Hochschule Baden-Würtemberg

PWA Progressive Web App

VPC Values Proposition Canvas

MVP Minimal Viable Product

API Application Programming Interface

IndexedDB Indexed Database API

API Application Programming Interface

HTML Hypertext Markup Language

1 Einleitung

1.1 Grundidee von Xpire

In Deutschland landen jährlich fast 13 Millionen Tonnen Lebensmittel in der Mülltonne. Runter gerechnet ergibt das 85,2 Kilogramm Essen pro Jahr pro Haushalt. Zu diesen Ergebnissen kommt eine Studie aus dem Jahr 2005, durchgeführt von der Universität Stuttgart.¹ Ein Teil dieser enormen Verschwendungen ist zurückzuführen auf Landwirte, Lebensmittelverarbeiter, Handel und Gastronomie. Mehr als die Hälfte der Abfälle fällt jedoch in privaten Haushalten an. Eine Verbesserung dieser Situation ist nur erreichbar, wenn Verbraucher lernen weniger einzukaufen, Lebensmittel wie Obst, Gemüse und Brot richtig zu lagern und Reste nicht bedenkenlos wegzwerfen.²

Dank der Digitalisierung, die inzwischen fast alle Lebensbereiche durchdrungen hat, und der Tatsache, dass weit mehr als die Hälfte aller Deutschen ein Smartphone besitzen (Tendenz steigend), können neue Technologien helfen, dieses Problem zu lösen. Wir, eine Gruppe von Studierenden der DHBW Mannheim, sehen daher ein großes Potential in der Entwicklung einer mobilen Applikation zur Verwaltung gekaufter Lebensmittel. Mit unserer mobilen Applikation „Xpire“ wollen wir dem Problem der Lebensmittelverschwendungen entgegen wirken und jedem auf einfache Art und Weise zu mehr Nachhaltigkeit verhelfen. Die Grundidee besteht darin, Xpire als mobile Applikation möglichst simpel und benutzerfreundlich zu gestalten. Zu viele Funktionen und Features schrecken den Benutzer ab und führen schnell zu Verwirrung und Frustration. Die Hauptfunktionalität beschränkt sich daher auf:

- das Hinzufügen eines Produktes
- das Anzeigen aller hinterlegten Produkte mit Verfallsdatum
- das Benachrichtigen, falls die Haltbarkeit eines Produktes kurz vor dem ablaufen ist

Die App enthält auch weitere Funktionen wie das Hinzufügen von Produktbildern, das Löschen eines hinterlegten Produktes oder das Verändern der gespeicherten Informationen zu einem Produkt.

¹vgl. Universität Stuttgart 2012.

²vgl. Universität Stuttgart 2012.

Ziel ist es alle gekauften Lebensmittel problemlos über die App verwalten zu können und ortsunabhängig immer zu wissen, wie der Haltbarkeitsstand der einzelnen Produkte ist. So soll die App dabei helfen, Produkte innerhalb der Haltbarkeit zu verwerten und weniger Abfall zu produzieren. Dies schont nicht nur den Geldbeutel jedes Einzelnen, sondern trägt auch zu einer nachhaltigeren Lebensweise bei und hilft somit verantwortungsbewusster mit den endlichen Ressourcen unserer Erde umzugehen.

1.2 Was ist eine PWA?

Bei der traditionellen Entwicklung mobiler Anwendungen war die Wiederverwendbarkeit von Code zwischen nativen Anwendungen, dem Web und mobilen Plattformen von Natur aus nicht gegeben. Dies ist auf die nicht kompatible Codebasis nativer Anwendungen zurückzuführen, was zu getrennten Projekten und Entwicklerumgebungen führt, wenn Unterstützung für mehrere Plattformen und Betriebssysteme gewünscht wird.³ Progressive Web Apps revolutionieren diesen Ansatz.

Als Webseite, die jegliche Eigenschaften einer nativen Applikation aufweist, kann sie auf dem Endgerät über den Webbrower installiert werden. Diese Eigenschaften beinhalten Installierbarkeit, Offlinefähigkeit oder Push-Benachrichtigungen, selbst wenn die Anwendung geschlossen wurde. Die progressive Web App stellt daher eine Mischung aus responsiver Webseite und nativen App dar. Diese plattformübergreifende Web Apps werden auf der Grundlage von HTML5, CSS3 und JavaScript erstellt. Das HTTP-Protokoll zur Kommunikation zwischen Webclient und Webserver stellt dabei die grundlegende Bedingung zur Verwendung einer PWA dar. Die PWA enthält sogenannte *ServiceWorker*, welche Funktionalitäten wie Offlinefähigkeit, Caching als auch Push-Benachrichtigungen ermöglichen. Bei einem *ServiceWorker* handelt es sich um einen in JavaScript implementierten Proxy, der zwischen der Anwendung und dem Server geschaltet wird.⁴ Die Verwendung einer PWA bietet sowohl für Entwickler als auch für den Anwender viele Vorteile. Durch den Mobile-First-Ansatz wird garantiert, dass die Web Apps problemlos auf mobilen Geräten innerhalb des Browsers funktionieren. Somit ist die User Experience wie in einer nativen App und bietet viele bekannte Vorteile während die Entwicklung in vielen Fällen um ein Vielfaches schneller, einfacher und kostengünstiger ist als die einer nativen App.

³vgl. Keist, Benisch und Müller 2016.

⁴vgl. Hume 2018.

1.3 Vorgehensweise

Um so effektiv wie möglich zusammen zu arbeiten zu können, haben wir uns eine flexible Art der Aufgabenaufteilung überlegt. Dabei sind wir wie folgt vorgegangen: Zunächst haben wir gemeinsam in einer Brainstorming-Session überlegt, welche Funktionalitäten die Xpire-App beinhalten soll, wie sie aussehen soll und welche Technologien wir für die Umsetzung verwenden möchten. Anschließend hat sich jedes Gruppenmitglied selbstständig mit den Grundlagen einer PWA als auch mit der Softwarebibliothek *React* vertraut gemacht. In einer zweiten Session haben wir gemeinsam die Code-Basis für das Projekt aufgesetzt und spezifische Verantwortungsbereiche im Projekt definiert, zu denen sich jeder, je nach Stärken und Interessen, selbst zuordnen konnte. Dabei ist zu beachten, dass eine Person nicht zwangsläufig nur im eigenen Verantwortungsbereich agieren musste. Jeder durfte natürlich entsprechend seiner Stärken den Fokus setzen und darüber hinaus weiteren Aufgaben außerhalb seines Schwerpunktes nachgehen und die anderen Team-Mitglieder bei ihren Aufgaben unterstützen. Wichtig war uns, dass sich jeder im Team wohl fühlt, dem nachgehen kann, worauf er Lust hat und, ganz wichtig, durch Learning-by-Doing ganz viel Neues erlernen kann.

Die Schwerpunkte wurden wie folgt eingeteilt:

- **Design:** Verena, Andrea
- **App-Entwicklung:** Felix, Milena, Yvonne, Verena
- **Datenbank:** Fabio, Jonas
- **Projektmanagement:** Andrea

Freitags, an unseren offiziellen Meeting-Terminen, stellten wir einander vor, welche Aufgaben in der Zwischenzeit umgesetzt wurden, definierten neue Aufgaben und reflektierten über die bisherige Vorgehensweise. Unter der Woche vereinbarten wir je nach Bedarf individuell Termine mit dem gesamten Team oder trafen uns in Teilgruppen, um bestimmte Aufgaben umzusetzen. Ein Protokoll der Meetings liegt in unserem GitHub-Repository im Dokument „meetings.md“ und kann unter folgendem Link aufgerufen werden:

<https://github.com/felixwaage/Xpire/blob/master/meetings.md>

Das gesamte Projekt kennzeichnet sich durch eine iterative Arbeitsweise, sodass genug Platz für neue Ideen und Erweiterungen bleibt. Ziel war es, zunächst ein Minimal-Viable-Product (MVP) zu entwickeln, um ein funktionierendes Produkt liefern zu können, welches sich in zukünftigen Schritten einfach erweitern lässt.

2 Analyse

2.1 Value Proposition Canvas

Ein Value Proposition Canvas (VPC) stellt eine Ergänzung zum Business Model dar und ist ein Werkzeug, welches das Designen, Testen, Verwalten und Erstellen von Kundenwertversprechen unterstützt. „Think of the Value Proposition as a contract between the customer and your company where the customer “hires” your company to solve a problem.“ (Clayton Christensen) Mit dieser Formulierung bringt Christensen ziemlich genau auf den Punkt, was man unter einem VPC versteht. Es bildet gewissermaßen einen Vertrag zwischen Kunde und Unternehmen, indem es darum geht, ein Problem des Kunden zu lösen. Das VPC hilft nicht nur systematisch das Wertversprechen des Unternehmens herauszuarbeiten, es ist auch Basis für Marketing, Pricing und den späteren Erfolg am Markt.⁵ Aus einer Umfrage von Simon & Kucher aus dem Jahr 2014 geht hervor, dass 72% aller Produktinnovationen scheitern. Hauptgrund dafür ist die fehlende Wahrnehmung beim Kunden.⁶

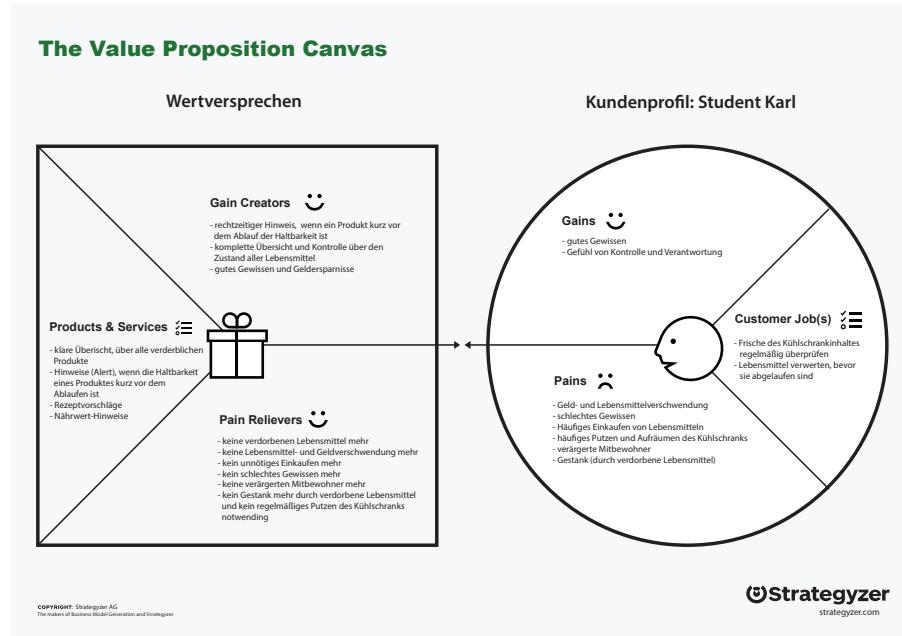


Abbildung 2.1: Value Proposition Canvas

⁵vgl. Pokorná et al. 2015.

⁶vgl. Simon & Kucher 2014.

Die Abbildung 2.1 zeigt den Value Proposition Canvas für die Xpire-App. Dabei stehen sich Wertversprechen der App und das Kundenprofil gegenüber. Die Kundenseite gliedert sich in „Customer Jobs“, „Gains“ und „Pains“. Dadurch wird aufgezeigt, welche Aufgaben der Kunde erledigen möchte, warum er diese erledigen möchte und mit welchen Herausforderungen und Schwierigkeiten er dabei konfrontiert wird. Auf der linken Seite des VPC sind die Wertversprechen des Unternehmens zu sehen, unterteilt in „Gain Creator“, „Pain Relievers“ und „Products & Services“.

Im vorliegenden Beispiel ist der Kunde ein Student namens Karl, der in einer Wohngemeinschaft wohnt und daher einen Kühlschrank mit seinen Mitbewohnern teilt. Er geht häufig einkaufen, hat meistens Geldprobleme und einen schlechten Überblick über seine Lebensmittel, da er oft gestresst oder unterwegs ist. Zu seinen Jobs (im Haushalt) gehört das regelmäßige Überprüfen des Kühlschranks und das Verwerten der Lebensmittel, bevor sie abgelaufen sind. Das gelingt ihm meistens leider nicht, weshalb er regelmäßig verdorbene Lebensmittel wegwerfen muss. Dies ist nicht nur eine Form der Lebensmittel- sondern auch der Geldverschwendug und führt bei Karl regelmäßig zu Frust und einem schlechten Gewissen. Zusätzlich ärgern sich seine Mitbewohner über den Gestank im Kühlschrank, wenn mal wieder ein Produkt verschimmelt ist. Karl ärgert sich ebenfalls, denn er muss öfter als ihm lieb ist den Kühlschrank putzen. Er wünscht sich, weniger verschwenderisch zu leben und mehr Kontrolle über den Stand seiner Lebensmittel zu haben. Er möchte ein gutes Gewissen haben, wenn es um seine Finanzen geht und verantwortungsvoll im Umgang mit Nahrungsmitteln sein.

Die mobile Applikation „Xpire“ hilft Karl seine Ziele zu erreichen und sein Verhalten bezüglich der Lebensmittel- und Geldverschwendug zu verbessern. Xpire ermöglicht ihm eine schnelle und einfache Übersicht über den Stand aller seiner Lebensmittel. Zusätzlich erhält er rechtzeitig Hinweise, wenn eines seiner gekauften Produkte kurz vor dem Ablauf der Haltbarkeit steht. Karl hat somit die volle Kontrolle über all seine Lebensmittel und kann nun weniger verschwenderisch leben. Das schont nicht nur seinen Geldbeutel, auch sein Gewissen und seine Mitbewohner freuen sich. Gestank und verschimmeltes Essen im Kühlschrank gehören nun der Vergangenheit an. Mit Xpire können alle gekauften Produkte in der App per Barcode hinterlegt oder manuell eingegeben werden. Die Produktdaten werden anschließend in einer lokalen Datenbank gespeichert. Zu jedem Produkt können zusätzliche Informationen wie Menge, Einkaufsdatum und Verfallsdatum hinterlegt werden. Xpire zeigt dem Benutzer auf übersichtliche Art und Weise all seine hinterlegten Produkte an und benachrichtigt ihn per Push-Notifications, wenn eines seiner Produkte kurz vor dem Ablauf des Haltbarkeitsdatums steht.

2.2 Benutzerprofilanalyse

Im Rahmen der Benutzerprofilanalyse werden die Eigenschaften der Endanwender untersucht, um die Gebrauchstauglichkeit der Anwendung sicherzustellen. Eine wesentliche Rolle spielen dabei die persönlichen Merkmale, Fähigkeiten und Kenntnisse der Benutzer sowie deren Einordnung in ein bestimmtes Nutzungsverhalten.

Da die progressive Web App von möglichst vielen Nutzern verwendet werden soll, gibt es kaum Einschränkungen, was die Zielgruppe betrifft. Die Gemeinsamkeit der Nutzer liegt im Besitz eines Kühlschranks, dem selbstständigem Tägigen von Lebensmittel-Einkäufen sowie im Besitz eines Smartphones oder eines Computers. Daraus lässt sich ableiten, dass die Zielgruppe überwiegend volljährig ist und ein Smartphone oder einen Computer besitzt. Laut Statista besitzen aktuell 66,5 Mio Menschen in Deutschland ein Smartphone.⁷ Ca. 13 Mio Menschen in Deutschland sind Minderjährig, wobei nicht alle Minderjährigen ein Smartphone besitzen. Zieht man trotzdem die Anzahl aller Minderjährigen von der Anzahl der Menschen, die in Deutschland ein Smartphone besitzen, ab, kommt man immer noch auf eine Anzahl von mindestens 49 Mio potentiellen Nutzer. Das sind mehr als 50% aller deutschen Staatsbürger.

All diese Nutzer können einen völlig unterschiedlichen Erfahrungsschatz bezüglich der Nutzung von neuen Technologien aufweisen. Von der Rentnerin, die nur selten einen Computer verwendet bis hin zum zwanzigjährigen Studierenden, der selbstständig Web Apps programmiert, können die unterschiedlichsten Ausgangsvoraussetzungen gegeben sein. Es ist davon auszugehen, dass ein Großteil der Anwender eine durchschnittliche Technikaffinität aufweist, es aber auch Anwender gibt, die keinerlei oder kaum Erfahrungen im Umgang mit Web Apps aufweisen. Demzufolge sollte die Oberfläche gleichzeitig so konzipiert sein, dass sie auch für Laien intuitiv und verständlich ist. Ein strukturierter und klarer Aufbau könnten diesem Aspekt Rechnung tragen. Schließlich sei noch zu erwähnen, dass die Web App vorerst nur für Deutschland entwickelt wird. Zu einem späteren Zeitpunkt ist die Internationalisierung der Applikation durchaus denkbar, dafür bedarf es dann einer Möglichkeit zur Auswahl unterschiedlicher Sprachen.

⁷<https://de.statista.com/statistik/daten/studie/500579/umfrage/prognose-zur-anzahl-der-smartphonenu...>

2.3 Anforderungsanalyse

Eine zentrale Fragestellung bei der Konzeption von progressiven Web Apps ist die nach den Zielen der Anwendung. Welche Aufgaben soll der Benutzer erledigen können und welche qualitativen sowie technischen Rahmenbedingungen sollen dabei berücksichtigt werden? Nach diesen Fragen wurde eine Liste von funktionalen und nicht-funktionalen Anforderungen erstellt, welche sich auf die erste Version der Xpire-App, das Minimal Viable Product (MVP), beziehen:

2.3.1 Funktionale Anforderungen

Die funktionalen Anforderungen beschreiben die Funktionalitäten, welche die Anwendung bereitstellen soll:

- F1 **Produkte hinterlegen:** Der Kunde soll die Möglichkeit haben, gekaufte Produkte per Barcode-Scan oder manuell hinterlegen zu können.
- F2 **Produktinformationen hinterlegen:** Der Benutzer hat die Möglichkeit, folgende Informationen zu den Produkten zu hinterlegen: Titel, Anzahl, Einkaufsdatum und Gültigkeitsdatum.
- F3 **Produkte verwalten:** Die Produkte sollen dem Benutzer in übersichtlicher Weise angezeigt werden. Der Nutzer hat die Möglichkeit ein Produkt zu löschen oder die hinterlegten Informationen zu den Produkten jederzeit zu verändern.
- F4 **Bild hinterlegen:** Benutzer sollen die Möglichkeit haben, pro Produkt ein Bild hochladen zu können.
- F5 **Benachrichtigungen erhalten:** Der Benutzer soll eine Push-Benachrichtigung erhalten, wenn ein Produkt kurz vor dem Ablauf seiner Haltbarkeit steht.

2.3.2 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen betreffen die Umstände, unter denen die geforderten Funktionalitäten zu erbringen sind:

- NF1 **Usability-Ziele:** Die Usability (dt.: Gebrauchstauglichkeit) ist definiert als das Ausmaß, in dem ein Produkt in einem bestimmten Anwendungskontext genutzt werden kann, um bestimmte Ziele mit geringem Aufwand und hoher Zufriedenheit zu erreichen.⁸ Basierend auf den Ergebnissen der Benutzerpro-

⁸vgl. Balzert, Klug und Pampuch 2009, S.3 ff.

filanalyse wurde der Fokus bei der erstellten progressiven Web App auf die Erreichung folgender Kriterien gelegt:

NF1.1 Verständliche Informationsstruktur: Eine klare Informationsstruktur schafft die Voraussetzung für die intuitive Bedienbarkeit des Systems und kann damit grundlegend zu einer positiven Benutzererfahrung beitragen.⁹ Aus diesem Grund sollen bereitgestellte Informationen möglichst logisch strukturiert und leicht verständlich sein. Grundsätzlich sollten dem Anwender nur die im aktuellen Kontext benötigten Informationen präsentiert und Unwichtiges weggelassen werden.

NF1.2 Durchdachtes Navigationskonzept: Eine gut gestaltete Navigation erleichtert dem Benutzer die Orientierung und helfen ihm, die folgenden Fragen zu beantworten: Wo befindet sich mich, wie bin ich hier hergekommen und wohin kann ich von hier aus gehen?¹⁰

NF2 Technische Rahmenbedingungen: Aus den technischen Randbedingungen wird in der Regel die Wahl der einzusetzenden Technologien getroffen. Diese sind:

NF2.1 Physikalische Nutzungsumgebung: Einsatzort der PWA kann überall sein. Um eine optimale Lesbarkeit zu garantieren, sollte daher auf die richtige Verwendung von Farbkontrasten und Schriftarten geachtet werden.

NF2.2 Hardware: Als Endgerät kann sowohl ein Smartphone, als auch ein Tablet oder ein PC verwendet werden. Die PWA muss daher so konzipiert sein, dass sie auf die Eigenschaften des jeweils benutzten Endgeräts optimal reagieren kann.

2.4 Monetarisierungsstrategie

Grundsätzlich gibt es zahlreiche Möglichkeiten, um mit einer App Geld zu verdienen. Wichtig ist, im Vorfeld zu überlegen, wie die App auf dem Markt positioniert werden soll und welche Monetarisierungsstrategie am besten passt. Zu den drei bekanntesten Möglichkeiten zählen¹¹:

⁹vgl. Moser 2012, S.106 ff.

¹⁰vgl. Moser 2012, S.116 ff.

¹¹vgl. Wiesche et al. 2018.

- **Paid-App:** Dies ist nicht die einfachste Möglichkeit, um Geld mit einer App zu verdienen. Statt die App kostenlos im Google Play Store oder im Apple Store anzubieten, wird die App kostenpflichtig angeboten, sodass dem Nutzer einmalige Kosten beim Download der App entstehen.
- **In-App-Werbung:** Das Schalten von Werbung innerhalb der App ist eine beliebte Methode um den Umsatz anzukurbeln. Die Möglichkeiten sind sehr vielfältig und reichen von klassischen Werbebanner bis hin zu boomender Videowerbung.
- **Freemium:** Bei dieser Monetarisierungsstrategie ist der Download der App kostenlos, der User hat dann die Möglichkeit durch In-App-Käufe oder einem Abo neue Inhalte, neue Funktionen oder die Laufzeit einer Vollversion zu verlängern. Dafür fallen dem User dann zusätzliche Kosten an.

Neben den genannten Beispielen gibt es noch viele weitere Möglichkeiten zur Monetarisierung einer App. Wir haben die verschiedenen Szenarien genauer beleuchtet und sind zu folgender Auswertung gekommen:

Eine Paid-App kommt für uns nicht in Frage, da wir die Xpire-App ausschließlich kostenlos anbieten möchten. Unser Ziel ist es, möglichst viele User zu gewinnen um so für mehr Nachhaltigkeit und einen verantwortungsvolleren Umgang mit Lebensmitteln zu sorgen. Nur wenige Benutzer sind bereit, für eine App Geld auszugeben und die Gefahr, dass andere Anbieter eine ähnliche, kostenlose Alternative anbieten, ist hoch. Daher scheidet diese Strategie aus. Das Schalten von Werbung sehen wir als guten Kompromiss, um die App für den User kostenlos zu gestalten und trotzdem Umsatz zu generieren. Falls wir zukünftig Werbung schalten werden, darf diese jedoch weder unseriös, aufdringlich oder nervig wirken und muss gegen Bezahlung deaktiviert werden können. Das Anbieten von zusätzlichen In-App-Käufen oder das Abschließen eines Abos um auf neue Inhalte oder Funktionen zugreifen zu können, halten wir in unserem Fall jedoch für wenig sinnvoll.

Es sind zusätzliche Funktionen wie das Aufzeigen von passenden Angeboten der umliegenden Supermärkte oder das Vorschlagen von Rezepten basierend auf den hinterlegten Produkten geplant. Diese zusätzlichen Funktionen sollen aber keine kostenpflichtigen Features darstellen und ebenfalls kostenlos angeboten werden. Vielmehr halten wir die Kooperation mit verschiedenen Rezeptseiten wie Chefkoch.de oder restegourmet.de als auch die Kooperation mit diversen Supermärkten für sinnvoll. Darüber hinaus besteht die Möglichkeit die App teilweise durch Spenden zu finanzieren und kostenlose Infrastruktur bei verschiedenen Hosting-Anbietern anzufragen, denn einige Anbieter stellen gemeinnützigen Vereinen kostenlosen Speicherplatz zu Verfügung.

3 Konzeption und Entwurf

3.1 Eingesetzte Technologien

3.1.1 React

React ist eine JavaScript-Bibliothek zur Erstellung von Benutzeroberflächen, die 2013 von Facebook unter BSD-Lizenz veröffentlicht wurde. Zentrales Konzept von React ist die Komponenten-bezogene Architektur, die als Basis für leicht nachvollziehbaren und wartbaren Frontend-Code dient. Komponenten erlauben es, die Benutzeroberfläche in autarke, wiederverwendbare Einheiten aufzuteilen. Dabei kapseln sie Struktur (HTML), Aussehen (CSS) und Logik (JavaScript).¹²

Neben React stellen Angular oder vue.js vergleichbare Alternativen dar. Wir haben uns als Team jedoch bewusst für die Nutzung von React entschieden aus folgenden Gründen:

- ein Großteil des Teams hat bereits mehr Erfahrungen mit React, nicht aber mit Angular oder vue.js gesammelt → dieser Erfahrungsschatz ermöglichte einen schnellen Einstieg ins Projekt
- Es gibt bereits viele Projekte, die React in Verbindung mit einer PWA verwenden → lieferte Orientierung und half beim Verständnis
- die Anforderungen an Xpire beinhalten keine außergewöhnlichen Ansprüche, welche nicht mit React erfüllt werden könnten

3.1.2 Datenbank

Die Prämisse, die gesamte Anwendung light-weight und dezentral zu gestalten, wirkt sich vor allem auch auf die Datenhaltung aus. In der Regel werden bei webbasierten Anwendungen die Daten, die für spätere Verwendungen zur Verfügung stehen sollen, in einer Datenbank auf einem zentralen Rechner gespeichert. Für die Xpire-App wollten wir aber auf eine solche Zentralisierung verzichten, um zum einen die laufenden Kosten minimal zu halten und zum anderen den höchsten Datenschutz zu

¹²vgl. Zeigermann und Hartmann 2016.

gewährleisten, indem die Daten nur auf dem Gerät des Benutzers gespeichert werden. Bei nativen Apps wird hierfür ein Verzeichnis auf dem Gerätespeicher angelegt, in dem sämtliche Dateien und Datenbanken dauerhaft abgelegt werden können. Bei einer Progressive Web App (PWA) ist es allerdings nicht vorgesehen, auf dem Dateisystem des jeweiligen Geräts zu operieren, da es sich letztendlich um eine Webseite handelt.

Hier bieten moderne Browser einige Möglichkeiten an, Daten lokal zu speichern. Die üblichen Möglichkeiten hierfür sind Cookies, SessionStorage und LocalStorage. Teilweise sind diese jedoch zeitlich, aber vor allem strukturell begrenzt und können so nicht als Ersatz für eine Datenbank herhalten. Abhilfe schaffen kann da die sogenannte IndexedDB, eine Schnittstelle des Browsers für eine primitive Datenbank. So ist es einer Webseite möglich, Daten lokal und nicht serverseitig in einer Datenbank zu speichern. Eine solche Datenbank erlaubt auch Hochleistungssuchen der Daten durch die Verwendung von Indizes.¹³

Um den Zugriff auf die IndexedDB zu vereinfachen nutzen wir die Wrapper-Bibliothek Dexie.¹⁴ Dexie löst drei Hauptprobleme mit der nativen IndexedDB-API: Mehrdeutige Fehlerbehandlung, schwache Anfragen und Code-Komplexität und bietet so eine saubere Datenbank-API, die für unseren Einsatz bestens performt. Angelegte Produkte können hiermitpersistiert, abgefragt und geändert werden. Neugierige Nutzer der Xpire-App können über die Entwicklerwerkzeuge ihres Browsers den Inhalt der indizierten Datenbank und somit die gespeicherten Produkte einsehen.

3.1.3 Platform

Für Xpire haben wir ein Repository auf GitHub angelegt, sodass wir die Web App kostenlos über GitHub Pages hosten können. GitHub Pages sind öffentliche Webseiten, die kostenlos über GitHub gehostet werden. GitHub-Benutzer können sowohl persönliche Webseiten als auch Webseiten, die sich auf bestimmte GitHub-Projekte beziehen, erstellen und hosten.¹⁵ Wir haben GitHub Pages aus folgenden gründen für unser Projekt gewählt:

- Es ist kostenlos und ermöglicht ein einfaches Deployment.
- Es ist zentral mit dem GitHub Repository verbunden.
- Es unterstützt die gemeinsame Entwicklung.

¹³https://developer.mozilla.org/de/docs/Web/API/IndexedDB_API

¹⁴<https://dexie.org>

¹⁵<https://pages.github.com/>

Da sich die Xpire-App aktuell in der Entwicklungsphase befindet, ist GitHub Pages eine sehr zufriedenstellende Technologie. Im späteren Verlauf, wenn die App live gehen soll und für Endanwender verfügbar gemacht werden soll, ist GitHub Pages jedoch nicht mehr die geeignete Wahl für das Hosting. Mit Voranschreiten der Entwicklung werden wir daher entweder auf die SAP Cloud Plattform, Amazon AWS oder einen gemieteten Server, beispielsweise von IONOS oder Strato, ausweichen.

3.1.4 Sonstiges

Authentifizierung: Eine Authentifizierung ist derzeit nicht vorhanden und wird, je nach Entwicklung der Funktionalitäten und Bedarf, zu einem späteren Zeitpunkt ergänzt. Falls die technischen Möglichkeiten es zulassen, wollen wir die Web App jedoch ohne Authentifizierung anbieten, da wir keine Daten unserer Endbenutzer erheben möchten. Für uns stellt der Datenschutz eine höhere Priorität dar.

Präsentations- und Feedbackmöglichkeiten: Präsentations- und Feedbackmöglichkeiten sind in der aktuellen Version der App nicht vorhanden, da sie nicht zu den definierten Anforderungen unseres MVP's gehören. Im zukünftigen Live-Betrieb soll die App jedoch Feedbackmöglichkeiten für den Nutzer zur Verfügung stellen, um einen kontinuierlichen, benutzerorientierten Service anbieten zu können. Die Möglichkeiten zur Einholung des Feedbacks sollen den Nutzer in der Anwendung der App jedoch nicht einschränken. Daher sollen es keine willkürlich auftauchenden Bewertungs-Popups geben, sondern lediglich einen Feedback-Button, welcher den Nutzer zu einem Feedback-Formular navigiert. Die Abgabe von Feedback erfolgt auf freiwilliger Basis.

3.2 Konzeptuelles Modell

In den vorherigen Abschnitten wurden bereits einige Merkmale der Xpire-App beschrieben. Darauf aufbauend sollen im Folgenden die Zusammenhänge der verschiedenen Bestandteile näher erläutert werden und somit einen detaillierten Überblick über das konzeptionelle Modell liefern.

Kern der Xpire-App ist das Frontend, welches mit Hilfe von React als PWA konzipiert ist. Wie den Abbildungen 3.1 und 3.2 entnommen werden kann, besteht die Xpire-App aus 3 Ansichten: *Home-Screen*, *Product-Screen* und dem *Create-Screen*. Diese Ansichten werden in React durch Komponenten realisiert, welche in unterschiedlichen Kontexten wiederverwendet werden können. Der Home-Screen besitzt demzufolge die Komponente AppBar und eine Komponente zur Listendarstellung der Produkte. Die Ansichten *Product-Screen* und *Create-Screen* werden durch dieselbe Komponente realisiert, da zum Erstellen und Anzeigen der Produktinformation ähnliche UI-Bestandteile benötigt werden. Die Darstellung passt sich hierbei automatisch anhand der übergebenen Parameter an. Hierdurch können Code-Duplikate vermieden werden und auch der Raum für Fehler wird reduziert.

Ein Weiterer wichtiger Bestandteil der Anwendung ist die IndexedDB, welche zur dauerhaften Persistierung der Daten verwendet wird. Wie bereits in Abschnitt 3.1.2 erwähnt wird hierfür das Modul *Dexie* verwendet. Neben der Datenhaltung in der lokalen Datenbank wird eine aktuelle Kopie der gesamten Datenbank in einem Array im *State* der App-Komponente synchron gehalten, wodurch sowohl die Listen-Komponente als auch die Product-Screen-Komponente darauf zugreifen können.

Der Nutzer soll später seine Produkte mit Hilfe des einheitlichen Barcodes hinzufügen können. Dies soll die Verwendung deutlich beschleunigen und vereinfachen, da so Informationen wie Titel, Gewicht, usw. nicht manuell eingetragen werden müssen. Um die dazu benötigten Informationen zu erhalten, wird die OpenFoodFacts-API verwendet. Diese führt eine Datenbank mit umfassenden Produktinformationen und kann Anfragen über den Barcode entgegennehmen. Des Weiteren wird diese API verwendet, um Abbildungen der einzelnen Produkte zu erhalten. Dadurch wird dem Nutzer die Verwendung der Anwendung zusätzlich vereinfacht. Darüber hinaus ist in einer späteren Version angedacht, die Produktbilder lokal in der Datenbank zu speichern, um auch bei Offline-Betrieb die Bilder anzeigen zu können.

3.2.1 Notifications

Damit der Nutzer der App diese nicht aktiv öffnen muss, um über demnächst ablaufende Produkte informiert zu werden, ist es sehr sinnvoll, dass die App automatisch Push-Mitteilungen versendet. Moderne Browser können heutzutage solche Benachrichtigungen von Webseiten anzeigen, die hierfür die Notifications-API¹⁶ nutzen. Normalerweise werden solche Push-Nachrichten von einem Server der Webseite ausgelöst. Da unsere Anwendung aber komplett dezentral ausgeführt werden soll, muss eine Möglichkeit gefunden werden, die Notifications on-Device zu triggern. Es genügt nämlich nicht, bei bestimmten Interaktionen über die angebundenen Event-Handler eine Mitteilung auszulösen, sie müssen vielmehr terminiert werden können.

Hierzu gibt es im aktuellen Web-Standard keine vorgesehene Funktion, jedoch testet Google ganz aktuell dahingehend eine Erweiterung der Notifications-API.¹⁷ Im Rahmen eines sogenannten Origin Trials wird ein sicheres Experimentieren mit neuen Funktionen der Web-Plattform ermöglicht. Um die bestmöglichen Designs zu erzielen, werden solche neuen Funktionen nicht vorzeitig zu De-facto-Standards, auf die sich Webentwickler dann verlassen und eine Abänderung nur schwer möglich machen, sondern sie werden in einem iterativen Prozess getestet und durch Feedback verbessert, bevor sie letztendlich in den Standard ausgerollt werden. So sollte es im Idealfall einfacher sein, neue Funktionen freizulegen und zu iterieren, aber die Versuchspopulation zuverlässig einzuschränken. Mit einer Testpopulation von Entwicklern, die sich verpflichtet haben, Feedback zu geben, und Begrenzungen in der Größe der Anwenderbasis und der Versuchsdauer kann die Iteration schneller erfolgen, aber ohne das Risiko eines sogenannten Burn-in, also einer Resistenz gegen Veränderungen. Nun haben wir also ein zeitlich begrenztes Token von Google angefordert, um die Funktion *Notification Triggers*¹⁸ für unsere Webapp freizuschalten. Dieses wird im <head>-Tag der Hypertext Markup Language (HTML)-Seite mitgegeben. Zusätzlich muss, da wir React verwenden, dem Compiler per ESLint-Befehl mitgeteilt werden, dass Undef-Fehler für die verwendete Klasse `TimestampTrigger` ignoriert werden sollen, schließlich ist sie noch nicht Teil des Standards, wird aber von Chrome 80+ entsprechend erkannt. Mit dieser Beta-Funktion ist es dann möglich, einen Zeitstempel für das Feuern einer Mitteilung festzulegen und im Service-Worker zu registrieren.¹⁹ Somit ist es uns gelungen, für jedes Produkt einen Alarm zu setzen, der rechtzeitig vor Verfall den User per Push-Mitteilung informiert.

¹⁶https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API

¹⁷https://developers.chrome.com/origintrials/#/view_trial/6883752030435803137

¹⁸<https://github.com/rknoll/notification-triggers>

¹⁹<https://web.dev/notification-triggers/>

3.3 Entwicklung von Prototypen

3.3.1 Figma

Zur Erstellung der Prototypen haben wir das Design-Tool „Figma“ verwendet. Dies ist ein Kollaborations-Tool für Designer und funktioniert ähnlich wie Sketch oder Adobe XD, zeichnet sich aber durch zwei signifikante Unterschiede aus:²⁰

- Es läuft zu 100 % im Browser (keine Installation nötig)
- Möglichkeit der Kollaboration mit anderen Personen in Echtzeit

Dadurch liefert Figma viele Vorteile, die das gemeinsame Erstellen von Prototypen stark verbessern. Das Tool ist von Designern für Designer entwickelt wurden und einfach im handling. Benutzeroberflächen lassen sich schnell und effektiv designen, ohne viel Zeit in die Einarbeitung des Tools investieren zu müssen. Das kollaborative Arbeiten in Echtzeit ermöglicht die unkomplizierte Zusammenarbeit am selben Entwurf, ohne extra ein Tool zu installieren oder sich physisch treffen zu müssen. Des Weiteren bietet Figma die Möglichkeit, erstellte Screens in vier verschiedenen Formaten zu exportieren, als auch Interaktionen hinzuzufügen, um das Projekt im Live-Preview betrachten zu können. Im Live-Preview lassen sich alle zuvor definierten Interaktionen, wie beispielsweise das Navigieren auf einen anderen Screen, ausprobieren, sodass das Feeling einer echten Anwendung nahezu perfekt imitiert wird.

3.3.2 Mobile First

Da unsere Zielgruppe eine breite Masse an Menschen beinhaltet, haben wir Wert darauf gelegt, dass Xpire möglichst einfach und unkompliziert verwendet werden kann. Einem ansprechenden User Interface sowie einer intuitive User Experience werden daher besondere Bedeutung zugeschrieben.

Im Jahr 2015 meldete Google erstmals, dass mehr Suchanfragen über mobile Endgeräte als über Desktop-Geräte erfolgten. Seitdem steigt der Anteil der Internetnutzer, die auch mit dem Smartphone online gehen, rapide.²¹ Resultierend aus dieser Entwicklung ist die Wahl des Mobile-First-Ansatzes selbsterklärend. Inzwischen ist nicht mobil-optimierter (responsive) Content kaum noch vorstellbar. Mobile First

²⁰<https://www.figma.com>

²¹<https://de.statista.com/statistik/daten/studie/633698/umfrage/anteil-der-mobilen-internetnutzer-in-deutschland/>

lässt sich als neuer Denkansatz im Webdesign definieren. Das Design einer Website wird dabei erstmal in der mobilen Version optimiert, bevor es für größere Bildschirme entwickelt wird. Man arbeitet also von der kleinsten Layout-Version hin zur größten.²²

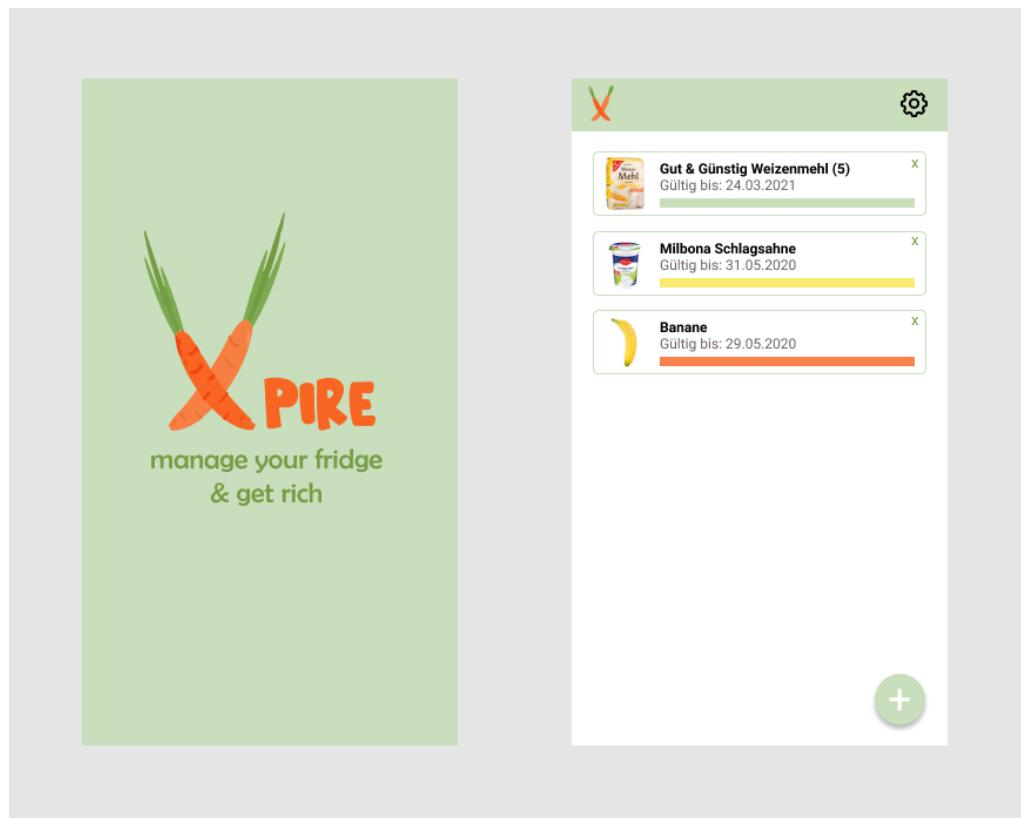


Abbildung 3.1: Xpire Welcome-Screen und Home-Screen

Die Abbildung 3.1 zeigt den *Welcome-Screen* und den *Home-Screen* der Xpire-App. Der *Welcome-Screen* zeigt das Xpire-Logo, welches mit dem vektorbasierten Grafik- und Zeichenprogramm „Adobe Illustrator“ erstellt wurde. Daher kann es in unterschiedlichen Formaten exportiert und schnell angepasst oder verändert werden. Die gewählte Farbkombination als auch die verwendete Symbolik verleihen dem Logo einen hohen Wiedererkennungswert und ein gewisses Öko-Flair. Das Motto „Manage your fridge & get rich“ ist nicht nur ein einprägsamer Reim, der sich gut als Werbespruch verwenden lässt, gleichzeitig stellt er eine Anspielung der monetären Ersparnisse dar, die man durch die Benutzung der Xpire-App erzielen kann.

Der *Home-Screen* in Abbildung 3.1 zeigt die aktuell hinterlegten Produkte mit Bild, Produktnamen, Verfallsdatum und einer farbigen Statusbar. Ist die Statusbar grün, bedeutet das, dass das Produkt noch haltbar ist und auch nicht unmittelbar vor dem

²²<https://www.interaction-design.org/literature/article/mobile-first-and-the-power-of-mobile-computing>

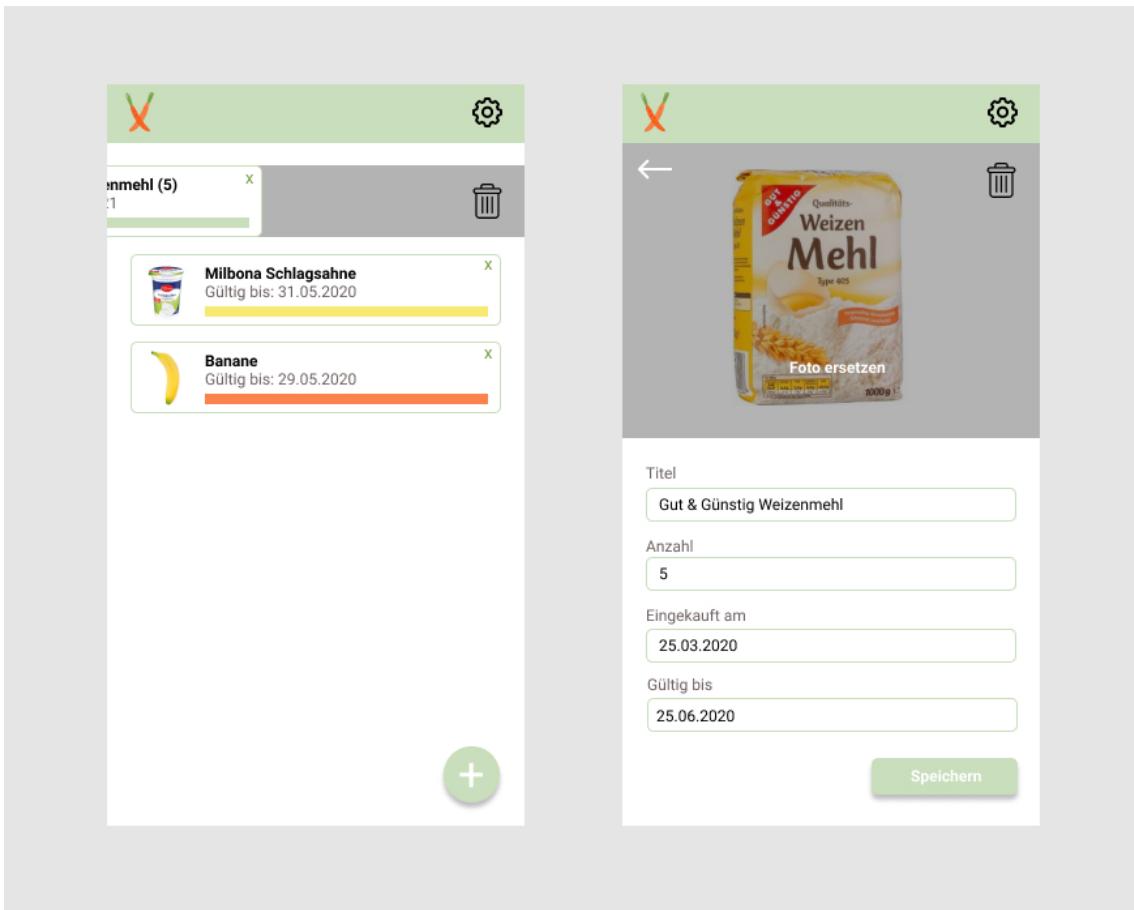


Abbildung 3.2: Xpire Delete-Screen und Product-Screen

Ablauf der Haltbarkeit steht. Ist der Balken gelb, wie im Mockup bei der „Milbona Schlagsahne“ bedeutet dies, dass das Produkt kurz vor dem Ablauf der Haltbarkeit steht. Wechselt ein Produkt in den gelben Status, erhält der Benutzer eine Push-Notification, mit dem entsprechenden Hinweis zum Haltbarkeitsablauf des jeweiligen Produktes. Ist der Balken rot gefärbt, hat das Produkt das entsprechende Verfallsdatum bereits erreicht oder sogar überschritten.

Der *Delete-Screen* zeigt, wie man ein Produkt durch wischen nach links löschen kann. Da sich die Anforderungen an die Bedienbarkeit in der Desktop-Version von der Smartphone-Version unterscheiden, gibt es zum Löschen eines Produktes in der Desktop-Version ein Müllheimer-Icon. Wird im *Home-Screen* ein Produkt angeklickt, so wird der Nutzer zum *Product-Screen* des jeweiligen Produktes geleitet. Diese Ansicht enthält Details zum Produkt, wie Name, Anzahl, Einkaufdatum und Verfallsdatum und erlaubt dem Nutzer ein Bild des Produktes zu hinterlegen. Alle hinterlegten Informationen lassen sich in dieser Ansicht vom Nutzer ändern.

4 Präsentation der PWA

In diesem Kapitel wird die umgesetzte PWA, die zuvor durch Mockups dargestellt wurde, vorgestellt. Da die Grundlagen zur Technologie bereits im Kapitel 3 ausführlich erläutert worden sind, wird hier nicht mehr näher auf die Implementierungsdetails eingegangen, sondern lediglich das Resultat anhand von Screenshots vorgestellt. Abschließend findet noch ein Abgleich mit den zuvor definierten Anforderungen statt.

Die Umsetzung der PWA erfolgte mithilfe der JavaScript-Bibliothek React auf Basis des entwickelten Prototypen. Im Folgenden wird der aktuelle Entwicklungsstand der App näher vorgestellt. Dabei wird auf die Gestaltung und die Navigation als auch auf die Funktionalität der PWA eingegangen.

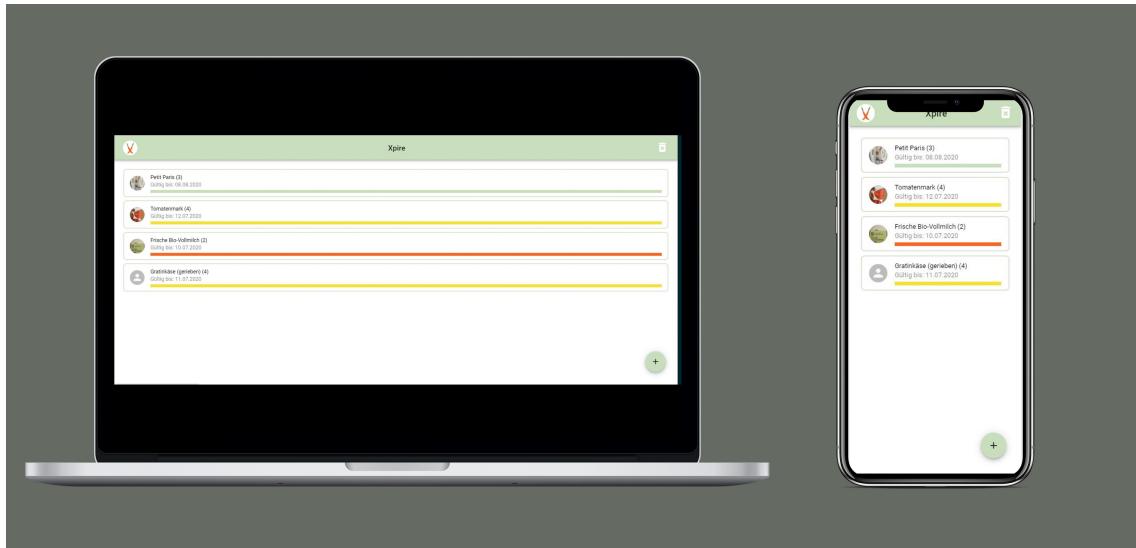


Abbildung 4.1: Xpire auf Laptop vs. Smartphone

Die Abbildung 4.1 zeigt, wie sich die Xpire-App verschiedenen Bildschirmgrößen problemlos anpasst. Zu sehen ist der *Home-Screen* mit einer Auswahl an Beispiel-Produkten sowohl auf dem Laptop als auch auf dem Smartphone.

Um Xpire auf dem Laptop zu installieren, muss man folgenden Link in die Adresszeile seines Browsers eingeben: <https://felixwaage.github.io/Xpire/>. Es erscheint die App, wie zu sehen in Abbildung 4.2 im Browser und lässt sich über ein „+“ ganz rechts in der Adresszeile als App dem Desktop hinzufügen. Nach einem Klick auf den

Plus-Button öffnet sich ein Pop-Up, welches es dem Nutzer ermöglicht, die Installation durchzuführen. Klickt man den Installations-Button, wird im dritten Schritt die Xpire-App auf dem Desktop hinterlegt, siehe Abbildung 4.2.

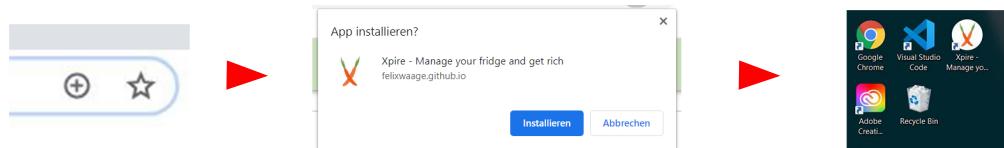


Abbildung 4.2: Xpire-Installationsprozess am Laptop

Die Installation auf dem Smartphone erfolgt äquivalent zu der auf dem Desktop. Nach dem Öffnen des Links wird über ein Pop-Up die Möglichkeit zur Installation bereit gestellt. Xpire wird dann auf dem Smartphone angezeigt wie eine native mobile App und weist auch nach dem Öffnen keine Unterschiede zu einer nativen App auf. Dennoch läuft die App als PWA komplett im Browser.

Damit die User Experience der PWA-Applikation auf dem Smartphone ähnlich zu einer nativen Applikation ist, wird ein sogenannter *Splash-Screen* implementiert. Dieser erscheint nachdem der Benutzer die auf dem Smartphone gespeicherte Xpire-PWA öffnet. In Abbildung 4.4 auf der nächsten Seite werden die Splash-Screens abgebildet, wobei dieser bei einem iOS-Betriebssystem in weiß und für Android-Geräte in grün erscheint.

Der Splash Screen sowie weitere Funktionalität der Xpire-Applikation sind in einer Bildschirmaufnahme mit dem Betriebssystem iOS zu sehen, welche unter folgendem Link verfügbar ist:

<https://github.com/felixwaage/Xpire/blob/master/iOS-Bildschirmaufnahme.mov>

Push-Benachrichtigungen mit Hinweisen zum Ablauf der Haltbarkeit hinterlegter Produkte werden dem User wie in Abbildung 4.3 angezeigt.

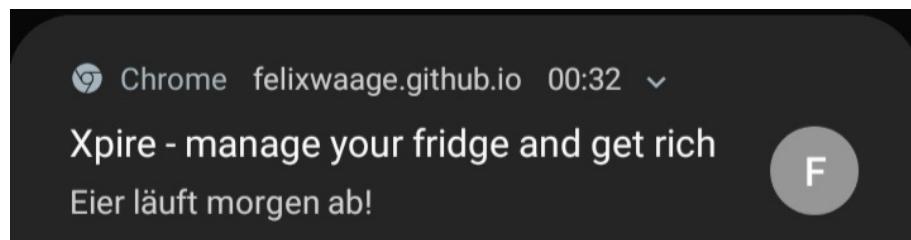


Abbildung 4.3: Push-Notification



Abbildung 4.4: Splash Screen für iOS und Android

Die Gestaltung als auch der User-Flow der entwickelten PWA entsprechen fast eins zu eins dem der Mockups. Geringe Unterschiede lassen sich an folgenden Punkten feststellen:

- **AppBar:** Das Logo in der entwickelten PWA hat einen weißen Kreis als Hintergrund. Das Icon oben rechts ist nicht wie im Mockup ein Zahnrad, sondern ein weißes Mülleimer-Icon.
- **Produkt löschen:** Die Produkte lassen sich einzeln nicht direkt im *Home-Screen* per Wischen nach links, wie designed im Mockup, löschen, sondern lediglich im *Produkt-Screen* über ein Mülleimer-Icon. Wird jedoch das Mülleimer-Icon oben rechts in der AppBar des *Home-Screen* dar, erhält der Nutzer die Möglichkeit, alle hinterlegten Produkte auf einmal zu löschen.
- **Produkte verwalten:** Der *Produkt-Screen* wurde implementiert, so wie er im Mockup designed wurde. Das hinterlegte Bild wird angezeigt und Name, Anzahl, Einkaufsdatum und Ablaufdatum lassen sich durch das Anklicken des jeweiligen Textfeldes ändern. Lediglich die Terminologie des Buttons wurde von „Speichern“ zu „Ändern“ verändert.

Neben der Umsetzung des erstellten Mockups, wurde zusätzlich eine Ansicht zum Hinzufügen eines neuen Produktes entwickelt, zu sehen in Abbildung 4.5. Der Nutzer hat die Möglichkeit einen Barcode zu scannen oder ihn manuell im Barcode-Textfeld einzugeben. Über die OpenFood-API wird automatisch das jeweilige Produktbild angezeigt und der Titel im dazugehörigen Feld angezeigt.

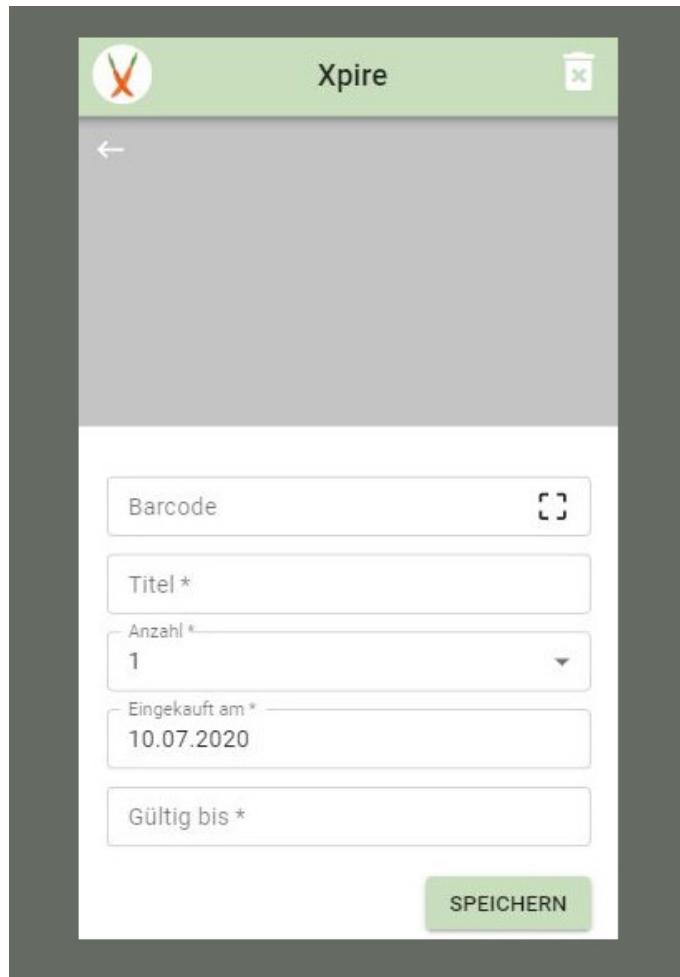


Abbildung 4.5: Produkt hinzufügen

Die Standard-Anzahl der Produkte beträgt 1, lässt sich aber über ein Dropdown einfach erhöhen. In dem Eingabefeld für das Einkaufsdatum ist automatisch das aktuelle Datum vorausgewählt, was sich jedoch ändern lässt, sobald man in das Textfeld reinklickt. Es öffnet sich ein Date-Picker, genau wie bei der Eingabe des Gültigkeitsdatums. Nach dem Speichern wird das Produkt automatisch dem *Home-Screen* hinzugefügt.

4.1 Anforderungsabgleich

In diesem Abschnitt wird betrachtet, welche Vorgaben der Spezifikation aus Kapitel 2.3 realisiert werden konnten. Die Bewertung des Erfüllungsgrads sowie die Beschreibung beziehen sich hierbei lediglich auf den aktuellen Stand der Implementierung.

4.1.1 Abgleich der funktionalen Anforderungen

Nr.	Erfüllt	Beschreibung
F1	Ja	Auf der Ansicht <i>Product-Screen</i> hat der Nutzer die Möglichkeit, gekaufte Produkte per Barcodenummer einzugeben. Daraufhin wird automatisch das dazugehörige Produktbild angezeigt.
F2	Ja	Der App-User kann im <i>Product-Screen</i> nicht nur das Bild hochladen und ändern, er hat auch die Möglichkeit alle hinterlegten Produktinformationen (Name, Anzahl, Einkaufsdatum, Verfallsdatum) zu ändern.
F3	Ja	Auf dem <i>Home-Screen</i> werden dem Benutzer alle hinterlegten Produkte übersichtlich angezeigt und er hat die Möglichkeit diese zu verwalten, also Produkte zu löschen oder Produktinformationen zu verändern.
F4	Teils	Der Benutzer kann manuell kein Bild hochladen. Jedoch wird ihm ein Bild seines Produktes angezeigt, sobald er den Barcode einscannt oder den Code manuell in das dafür vorgesehene Eingabefeld auf dem <i>Product-Screen</i> eingibt.
F5	Ja	Ist ein Produkt nur noch 3 Tage haltbar, erhält der User eine Push-Notification, um an das zeitnahe Verbrauchen dieses Produktes erinnert zu werden.

4.1.2 Abgleich der nicht-funktionalen Anforderungen

Nr.	Erfüllt	Beschreibung
NF1.1	Ja	Informationen wurden möglichst sinnvoll strukturiert, bezeichnet und dargeboten.
NF1.2	Ja	Die Navigation wurde so konzipiert, dass sie dem Benutzer Orientierung auf jeder Ansicht innerhalb der App bietet. Durch unterschiedliche Farbgebungen werden auf visuelle Art und Weise zusätzlich Informationen zum Status der Haltbarkeit eines Lebensmittels vermittelt.
NF2.1	Ja	Zur Erhöhung der Lesbarkeit wurde auf eine kontrastreiche Darstellung geachtet und eine gut leserliche und Browser-kompatible Schriftart gewählt.
NF2.2	Ja	Die PWA passt sich an die geforderten Bildschirmgrößen an. Durch den Mobile-First-Ansatz wirkt sie optisch wie eine native mobile Applikation, passt sich aber auch problemlos der Bildschirmgröße eines Tablets oder Desktops an.

Um die einzelnen Funktionalitäten der PWA zu testen, haben wir regelmäßig den neusten Stand sowohl auf dem Smartphone als auch in der Desktop-Version überprüft. Da der Umfang der PWA noch recht überschaubar ist, war ein manuelles Testen der einzelnen Funktionen völlig ausreichend und zufriedenstellend. Bei Weiterentwicklung der PWA sollten jedoch verschiedene Test, wie beispielsweise Unit-Test, geschrieben und implementiert werden, um die Funktionalitäten der einzelnen Komponenten zu überprüfen.

Neben den manuellen Tests wurde der Code zum vorläufigen Abschluss des Projektes gerefactort, um die Verständlichkeit des Codes zu vereinfachen und eine problemlose Weiterentwicklung zu ermöglichen.

5 Zusammenfassung und Ausblick

Ziel war es, zunächst ein Minimal-Viable-Product (MVP) zu entwickeln, um ein funktionierendes Produkt liefern zu können, welches sich in zukünftigen Schritten einfach erweitern lässt. Dieses Ziel haben wir nicht nur erreicht, sondern sogar übertreffen können, da bereits Funktionalitäten, die erst für eine zweite Version definiert wurden, umgesetzt werden konnten. Zu diesen Funktionalitäten zählen beispielsweise das Scannen des Barcodes zur Produktidentifikation und die Anzeige des Status eines Produktes. Die Umsetzung der PWA hat folglich wie geplant funktioniert, wodurch wir viele Vorteile hatten und auf das Entwickeln für mehrere Plattformen verzichten konnten.

Zukünftige Schritte, um Xpire weiter zu entwickeln, könnten sein:

- Anbindung einer Rezepte-API, sodass dem Benutzer Vorschläge zu Rezepten entsprechend der hinterlegten Produkte erhalten kann
- Hinterlegen von Standardbildern und Standard-Verfallsdaten für Obst und Gemüse
- die Liste der hinterlegten Produkte sortiert anzeigen und verschiedene Sortierungsmöglichkeiten zur Verfügung stellen
- zusätzliche Informationen zu den Produkten anzeigen, wie beispielsweise Nährwerte oder Allergenkennzeichnungen
- Synchronisation über mehrere Geräte hinweg ermöglichen
- Automatische Erkennung des Mindesthaltbarkeitsdatum
- Einrichtung einer Mitgliederverwaltung
- Tipps für die Lagerung automatisch anzeigen
- Feedbackmöglichkeiten für den User
- Produktbilder lokal in Datenbank speichern
- verschiedene Tests (z.B. Unit-Tests)

Aktuell kann die App, wie bereits beschrieben, über den Link erreicht und auf einem beliebigen Endgerät installiert werden. Da das Konzept einer PWA jedoch noch nicht weit verbreitet ist und viele der Endanwender kein Fachwissen besitzen, vermuten wir, dass es vereinzelt zu Akzeptanz- und/oder Anwendungsproblemen seitens der Endanwender kommen kann. Daher halten wir eine zusätzliche Vermarktung über

den App- bzw. den Google Play Store für sinnvoll, da Endanwender hauptsächlich diese Plattformen verwenden, um neue Apps zu downloaden. Falls diese Form der Vermarktung tatsächlich realisiert wird, ist es erforderlich, rechtliche Vorgaben einzuhalten und eine Datenschutzerklärung als auch ein Impressum zu erstellen.

Abschließend wollen wir noch auf die Learnings eingehen, die wir während des Projektes gesammelt haben.

Learnings bezüglich PWA:

Während der Entwicklung sind einige Eigenarten von PWAs ans Licht getreten. Zunächst ist beim Testen immer daran zu denken, dass der jeweilige Browser eine Version der App im Cache behält und diese anzeigt, anstatt die Seite komplett neu zu laden. Für sinnvolles Testen musste daher immer der Browser-Cache inklusive Websitedaten gelöscht werden. Bei diesem Schritt wird auch die Indexed Database API (IndexedDB) geleert. Besonders bei kleinen Änderungen stellte sich dies als mühselig heraus. Eine Möglichkeit, dem vorzubeugen, wäre Service-Worker-Versioning einzuführen, was für uns zunächst aber zu viel Aufwand darstellte. Weiterhin ist bei der Entwicklung aufgefallen, dass speziell bei Funktionen von PWAs die Unterschiede der Browser hinsichtlich Kompatibilität noch deutlicher werden, als sie uns ohnehin schon bekannt waren. Meistens hat der Chrome-Browser hier die Nase vorn und setzt auch als erster neue Schnittstellen um, wie am Origin Trial der Notification Triggers erkennbar ist. Doch selbst bei Verwendung des gleichen Browsers kann es je nach Betriebssystem (Windows, macOS, Android) sein, dass manche Funktionen anders beschaffen oder schlicht nicht verfügbar sind. Ein Beispiel sind auch hier die Push-Benachrichtigungen, die letzten Endes den Restriktionen des Betriebssystems unterliegen. Auch bei der Darstellung des App-Icons sind solche Eigenarten zu beobachten.

technische Learnings:

- Implementierung eines übergeordneten Material UI-Themings
- Verknüpfung des UIs mit der IndexedDB
- besserer Umgang mit GitHub (Branching, Merging)
- Back-End wird nicht unbedingt benötigt, Daten können im Browser gespeichert werden
- Kenntnisse zum Arbeiten mit React gewonnen

Learnings bezüglich Teamarbeit:

- sinnvolle Verteilung der Aufgaben nach den jeweiligen Stärken

- gegenseitige Unterstützung, wenn es Probleme gab → gute Gruppendynamik
- Überlegungen vor der Entwicklung mit Value Proposition Canvas
- Regelmäßige Meetings sind zwar zeitaufwändig, aber helfen enorm für den Projektverlauf (immer auf dem aktuellen Stand, woran gearbeitet wird und bei wem es gerade Probleme gibt)

weitere Learnings:

- durch schnelles Deployment mit GitHub-Pages wird Vorteil von PWAs deutlich → einfaches Bereitstellen sowohl für iOS als auch Android
- gute Vorbereitung mit Mockups, Release-Pläne und Priorisierung zahlen sich im Projektverlauf aus
- teilweise schwieriger gute Tutorials für PWA bezogene Themen zu finden wie Offline Zugriffe, Push Benachrichtigungen, Handyzugriff, App öffnen von anderer App aus, auf lokale Dateien zugreifen, IndexedDb → native, spezifische Funktionen des Smartphones

Fazit:

Aufgefallen ist uns, dass es einige Sachen gibt, die für PWAs noch nicht verfügbar sind. Außerdem agiert der Softwareentwickler Apple noch eher unkooperative bezüglich PWAs.

Auch wenn es bei der Umsetzung teilweise noch Schwierigkeiten gibt und die PWA in ihrer Funktionalität noch nicht so mächtig wie native Apps ist, halten wir das Grundkonzept für sehr gut. Die Einarbeitung in das Thema PWA lohnt sich! Man muss nur ein Entwicklungsstack beherrschen z.B. React, JavaScript, NodeJS und hat keinen Mehraufwand, um die App auf vielen Plattformen anbieten zu können.

Kurzgefasst lässt sich festhalten, dass wir durch das Projekt sehr viel (Neues) lernen konnten, nicht nur bezüglich PWA und der eingesetzten Technologien. Auch haben wir gelernt eine strukturierte Arbeitsweise als Team umzusetzen, gemäß unserer Stärken verschiedenen Aufgaben nachzugehen und bei Bedarf einander zu helfen. Die Kommunikation als auch die Kooperation im Team hat wunderbar funktioniert und wir gehen mit vielen neuen, wertvollen Erfahrungen aus diesem Projekt heraus.

Literatur

- Balzert, Heide, Uwe Klug und Anja Pampuch (2009). *Webdesign & Web-Usability: Basiswissen für Web-Entwickler*. 2. Aufl. Informatik. Herdecke: W3L-Verl. ISBN: 978-3-86834-011-2.
- Hume, David (2018). *Progressive web apps*. Shelter Island, NY: Manning Publications. ISBN: 978-1-61729-458-7. URL: <http://proquest.tech.safaribooksonline.de/9781617294587>.
- Keist, Nikolai-Kevin, Sebastian Benisch und Christian Müller (2016). „Möglichkeiten und Grenzen der plattformübergreifenden App-Entwicklung“. In: *Mobile Anwendungen in Unternehmen*. Hrsg. von Thomas Barton, Christian Müller und Christian Seel. Wiesbaden: Springer Fachmedien Wiesbaden, S. 109–119. ISBN: 978-3-658-12010-8. DOI: 10.1007/978-3-658-12010-8\$\\backslash\$textunderscore. URL: https://doi.org/10.1007/978-3-658-12010-8_8.
- Moser, Christian (2012). *User Experience Design: Mit erlebniszentrierter Softwareentwicklung zu Produkten, die begeistern*. X.media.press. Berlin, Heidelberg: Springer. ISBN: 978-3-642-13362-6. DOI: 10.1007/978-3-642-13363-3. URL: <http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10582569>.
- Pokorná, Jitka et al. (2015). „Value Proposition Canvas: Identification of Pains, Gains and Customer Jobs at Farmers’ Markets“. In: *AGRIS on-line Papers in Economics and Informatics* 665-2016-45080, S. 8. DOI: 10.22004/ag.econ.231899. URL: <http://ageconsearch.umn.edu/record/231899>.
- Simon & Kucher (2014). *Press Release: 72 percent of all new products flop*. Hrsg. von Simon Kucher & Partner, Strategy & Marketing Consultants. URL: https://www.simon-kucher.com/sites/default/files/simon-kucher_global_pricing_study_2014.pdf.
- Universität Stuttgart (2012). *Ermittlung der weggeworfenen Lebensmittelmengen und Vorschläge zur Verminderung der Wegwerfrate bei Lebensmitteln in Deutschland: Kurzfassung*. Hrsg. von Bundesministerium für Ernährung, Landwirtschaft und Verbraucherschutz. URL: https://www.bmel.de/SharedDocs/Downloads/DE/_Ernaehrung/Lebensmittelverschwendungs/Studie_Lebensmittelabfaelle_Kurzfassung.pdf?__blob=publicationFile&v=3.

Wiesche, Manuel et al., Hrsg. (2018). *Management digitaler Plattformen: Konzeption und Realisierung eines offenen Ökosystems für intelligente Mobilitätsdienste in der Smart City*. Wiesbaden: Springer Fachmedien Wiesbaden. ISBN: 978-3-658-21214-8.

Zeigermann, Oliver und Nils Hartmann (2016). *React: Die praktische Einführung in React, React Router und Redux*. 1. Aufl. s.l.: dpunkt. ISBN: 3864919649. URL: <http://proquestcombo.safaribooksonline.com/9781492019459>.