Plenty Plants – Backend

|  |  |
| --- | --- |
| Autor: | Rudi Loderer |
| Modul: | Webtechnologien |
| Datum: | 06.07.2019 |

Inhalt

[System-Architektur 2](#_Toc13321190)

[Konsumierte Schnittstellen 2](#_Toc13321191)

[Angebotene Schnittstellen 2](#_Toc13321192)

[Datenbank 3](#_Toc13321193)

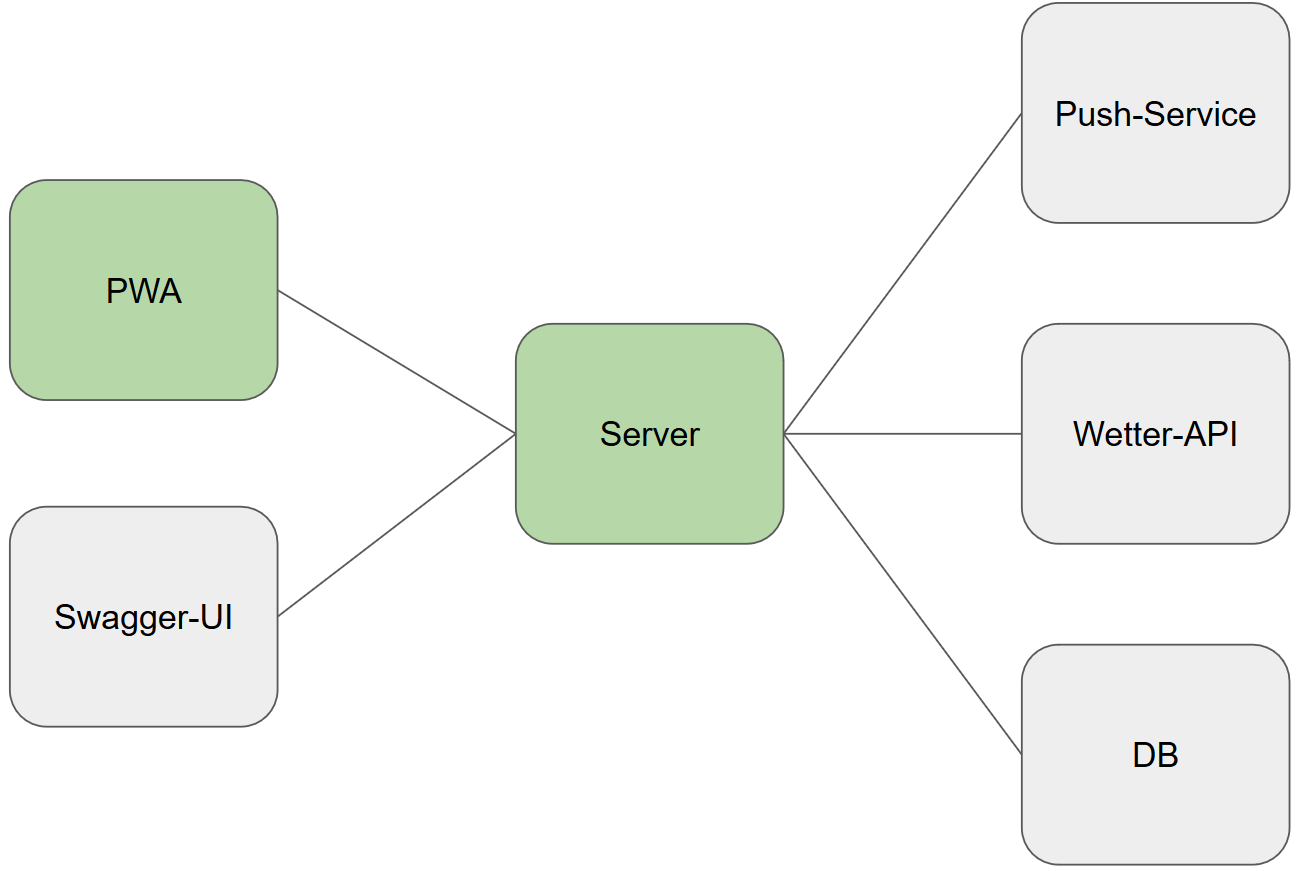
[Wetter-API 3](#_Toc13321194)

[Push-Benachrichtigungen 3](#_Toc13321195)

[Schnittstellen-Dokumentation 4](#_Toc13321196)

[Weitere Technologien & Frameworks 4](#_Toc13321197)

# System-Architektur



Bei der System-Architektur von „Plenty Plants“ handelt es sich um eine klassische Client-Server-Architektur. Unsere Progressive-Web-App (im Schaubild PWA genannt) nimmt dabei die Rolle des Clients ein. Diese Ausarbeitung beschäftigt sich mit dem Backend von „Plenty Plants“. Beim Backend handelt es sich um den Server-Anteil des Systems.

## Konsumierte Schnittstellen

Die Stammdaten von „Plenty Plants“ werden in einer Datenbank gehalten. Für Services, wie die Anzeige des aktuellen Wetters und Push-Benachrichtigungen bei Frost, werden Wetter-Daten benötigt. Diese Wetter-Daten werden von einer externen Wetter-API bezogen. Um Push-Benachrichtigungen auszuliefern, müssen zusätzlich die Push-Server der verschiedenen Browser angebunden werden.

## Angebotene Schnittstellen

Das Backend von „Plenty Plants“ bietet zwei APIs an. Die Hauptschnittstelle ist eine REST-API für den Client. Über diese Schnittstelle bezieht der Client alle benötigten Daten. Bei der zweiten Schnittstelle handelt es sich um eine Swagger-UI-Schnittstelle. Sie dient der Dokumentation der Hauptschnittstelle. Die Swagger-UI-Schnittstelle ist nicht für den Endbenutzer bestimmt. Sie hilft den Client-Entwicklern beim Anbinden der Hauptschnittstelle. Außerdem ermöglicht sie dem Backend-Entwickler das schnelle und unkomplizierte Testen des Servers.

# Datenbank

Die Wahl der Datenbank geschah unter zwei Hauptaspekten:

1. Es dürfen keine Kosten anfallen.
2. Der Verwaltungsaufwand ist minimal.

Beide Aspekte sprechen dafür, eine Datenbank als SAAS (Software as a Service) anzubinden. Letztendlich fiel die Wahl auf das Produkt „Tiny Turtle“ vom Provider „ElephantSQL“ (<https://www.elephantsql.com/plans.html>).

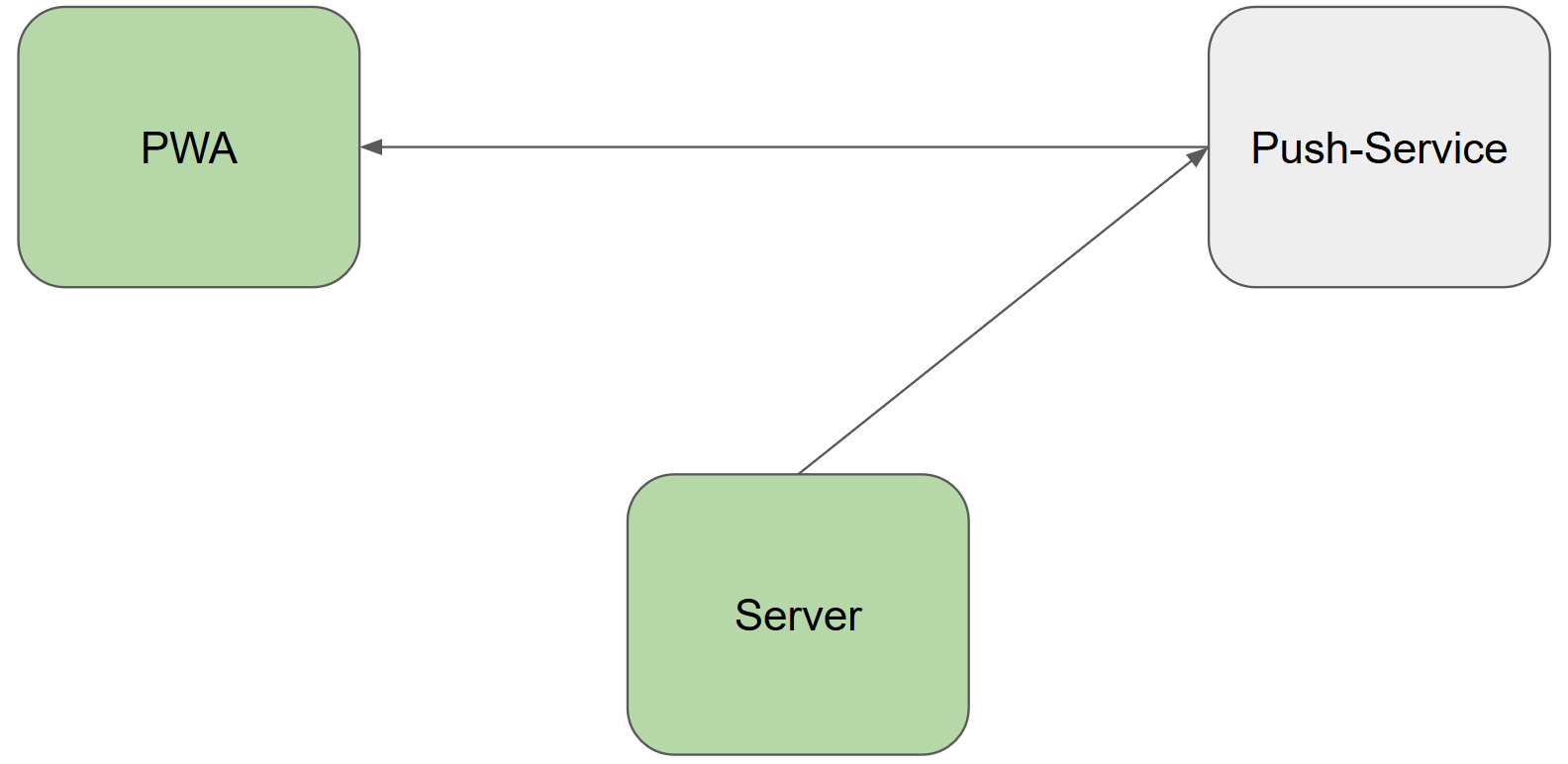
Das Produkt „Tiny Turtle“ ist kostenlos, jedoch ist die Anzahl gleichzeitiger Datenbankverbindungen und die Dateigröße der Datenbank eingeschränkt. Zum Ablegen unserer Stammdaten ist „Tiny Turtle“ vollkommen ausreichend. Grund dafür ist die geringe Menge an Stammdaten und Endbenutzern.

# Wetter-API

Die Wetter-Daten für „Plenty Plants“ werden nicht intern erhoben. Die Wetter-Daten werden von „OpenWeather“ bezogen. „OpenWeather“ bietet (unter anderem) kostenlose Schnittstellen für das aktuelle Wetter und eine Vorhersage für die nächsten fünf Tage (siehe <https://openweathermap.org/api>). Das aktuelle Wetter wird über eine entsprechende API an den Client weitergegeben. Die 5-Tage-Vorhersage kann für eine Frostwarnung genutzt werden (mehr Informationen bezüglich der Frostwarnung im Kapitel „Push-Benachrichtigungen“).

# Push-Benachrichtigungen

Der einzige Verwendungszweck für Push-Benachrichtigungen ist die Frostwarnung. Die Frostwarnung informiert den Endnutzer, falls in den folgenden 24 Stunden Frost vorhergesagt wird. So kann er rechtzeitig Gegenmaßnahmen ergreifen und seine Pflanzen schützen.



Push-Benachrichtigungen werden nicht direkt vom Server zum Client gesendet. Statt des direkten Wegs werden sie über einen sogenannten Push-Service zugestellt. Der Server legt die Benachrichtigung auf dem Push-Service ab. Der Client holt die Benachrichtigung dort ab und zeigt sie an. Wie genau dieser Vorgang im Detail abläuft ist nicht standardisiert. Verschiedene Browser implementieren den Vorgang unterschiedlich. Oftmals nutzen sie sogar verschiedene Push-Services.

Unsere prototypische Implementierung unterstützt die Browser Chrome und Firefox (siehe <https://developer.mozilla.org/de/docs/Web/API/Push_API#Browser_compatibility>).

Da es sich um einen Prototyp handelt wurde darauf verzichtet einen automatischen Trigger auf Basis der Wettervorhersage zu implementieren. Stattdessen muss die Frostwarnung über Swagger-UI ausgelöst werden (siehe <http://localhost:3000/api-docs/#/Notifications/SendFrostWarning>).

# Schnittstellen-Dokumentation

Die Schnittstellen-Dokumentation erfolgt mit Swagger. Angezeigt wird die Dokumentation unter <http://localhost:3000/api-docs/#/>. Diese Anzeige wird mithilfe des Frameworks swagger-ui-express (siehe <https://www.npmjs.com/package/swagger-ui-express>) ermöglicht. Swagger-ui-express übernimmt dabei die Aufgabe der Visualisierung. Die visualisierten Daten stammen aus einer Datei Namens „swagger.yaml“. Diese Datei kann man mit dem „Swagger Editor“ manuell erstellen. Im Fall von „Plenty Plants“ wird „swagger.yaml“ jedoch generiert. Hier kommt tsoa (<https://github.com/lukeautry/tsoa>) zum Einsatz. Tsoa generiert die Dokumentation auf Basis von Annotationen im Quellcode. So erlauben tsoa und Swagger das Erstellen einer aussagekräftigen Schnittstellendokumentation mit geringem Aufwand.

# Weitere Technologien & Frameworks

Das Backend von „Plenty Plants“ basiert auf einem „Node.js Express“-Server. Der Sourcecode des Servers ist in TypeScript verfasst. Die statische Typisierung in TypeScript ermöglicht bessere Unterstützung durch Entwicklungsumgebungen. Einerseits kann durch Autosuggestions die Programmiergeschwindigkeit erhöht werden. Andererseits werden Progammier-Fehler durch die Typprüfung in der Entwicklungsumgebung frühzeitig erkannt.

Für die Authentication wird auf eine Kombination der Frameworks Passport und bcrypt zurückgegriffen. Passport stellt die Authentication-Middleware, also den Großteil der Logik, der für Authentication notwendig ist. Bcrypt wird zum verschlüsseln der Passwörter genutzt. So können die Passwörter sicher in der Datenbank abgelegt werden.

Um den Implementierungsaufwand für Push-Benachrichtigungen gering zu halten wird zum Senden der Nachrichten zum Push-Service das Framework web-push verwendet. Dieses Framework kümmert sich darum, dass die Nachrichten Protokoll-Konform aufgebaut und der Inhalt korrekt verschlüsselt ist.

Die REST-API ist mit Bordmitteln des „Node.js Express“-Servers implementiert. Die Dokumentation der Schnittstelle erfolgt mit Swagger und tsoa (siehe <https://github.com/lukeautry/tsoa>). Für Details bezüglich der Schnittstelle-Dokumentation wird auf das entsprechende Kapitel verwiesen.