# Meilenstein 1

Firma: Doubleslash

Projekt: Fuhrparkmanagement System

Betreuer: Andreas Heinrich

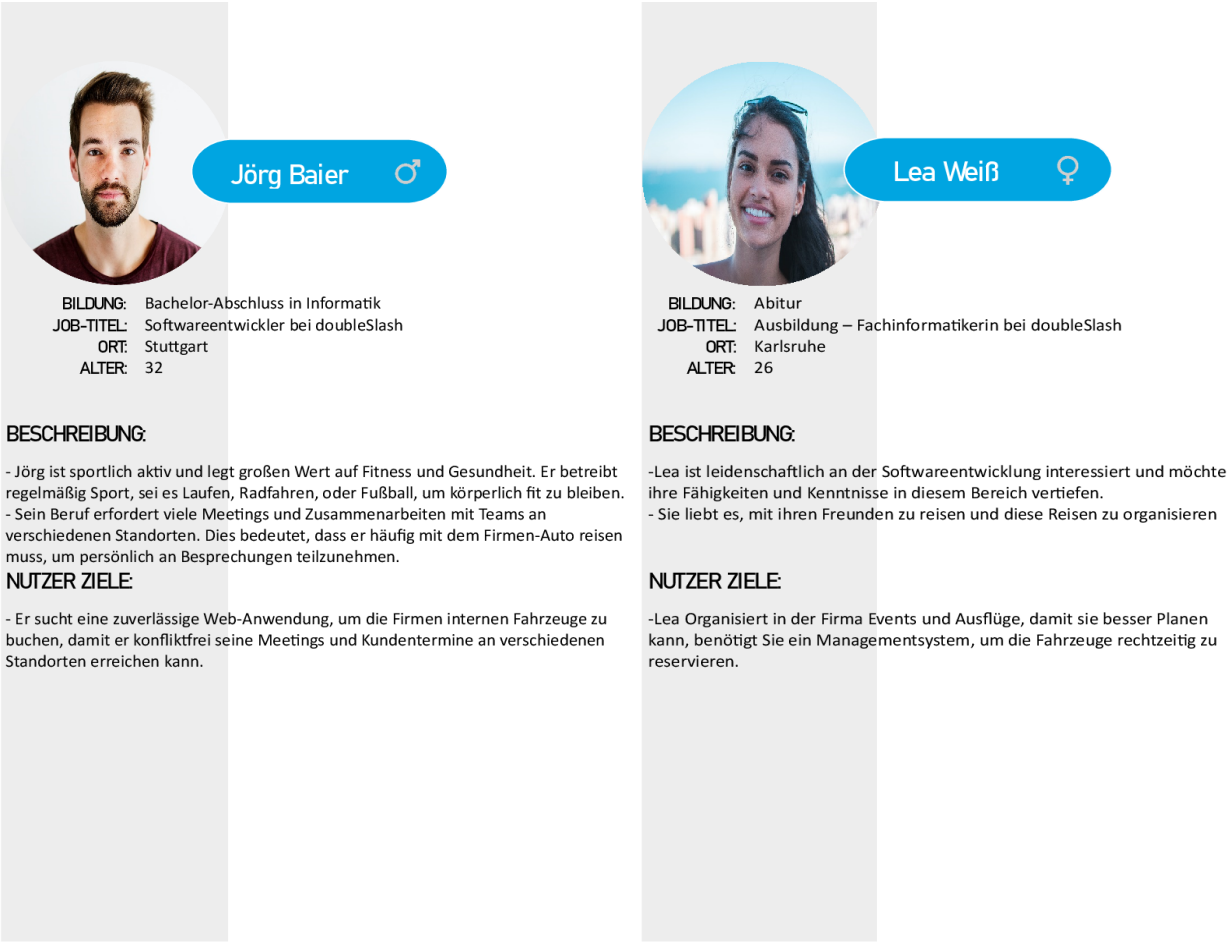
Kunde: Johannes Mayer

Team: Leonard Deininger, Okan Kizilagil, Cedric Schaaf, Nick Habermann, Edwin Starz

# Inhaltsverzeichnis

# Zielgruppe, Problem, Eigenschaften

Die Zielgruppe sind alle Mitarbeiter der Firma doubleSlash an den Standorten Stuttgart, Karlsruhe, München und Friedrichshafen.




**Problem**: Buchung der Fuhrpark Fahrzeuge ist nicht übersichtlich und nicht zentral mit einer Plattform, in der Vergangenheit wurden Fahrzeuge doppelt verbucht, da zum Teil über Outlook und Confluence parallel gebucht wurde.

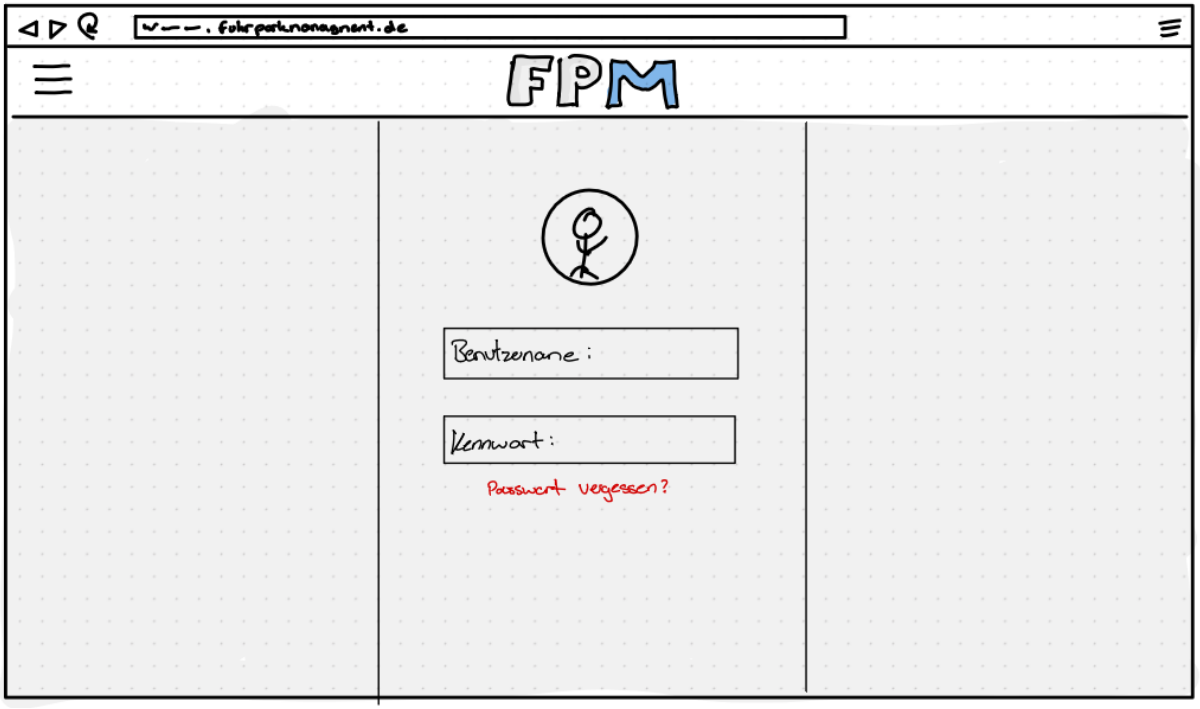
**Eigenschaften**

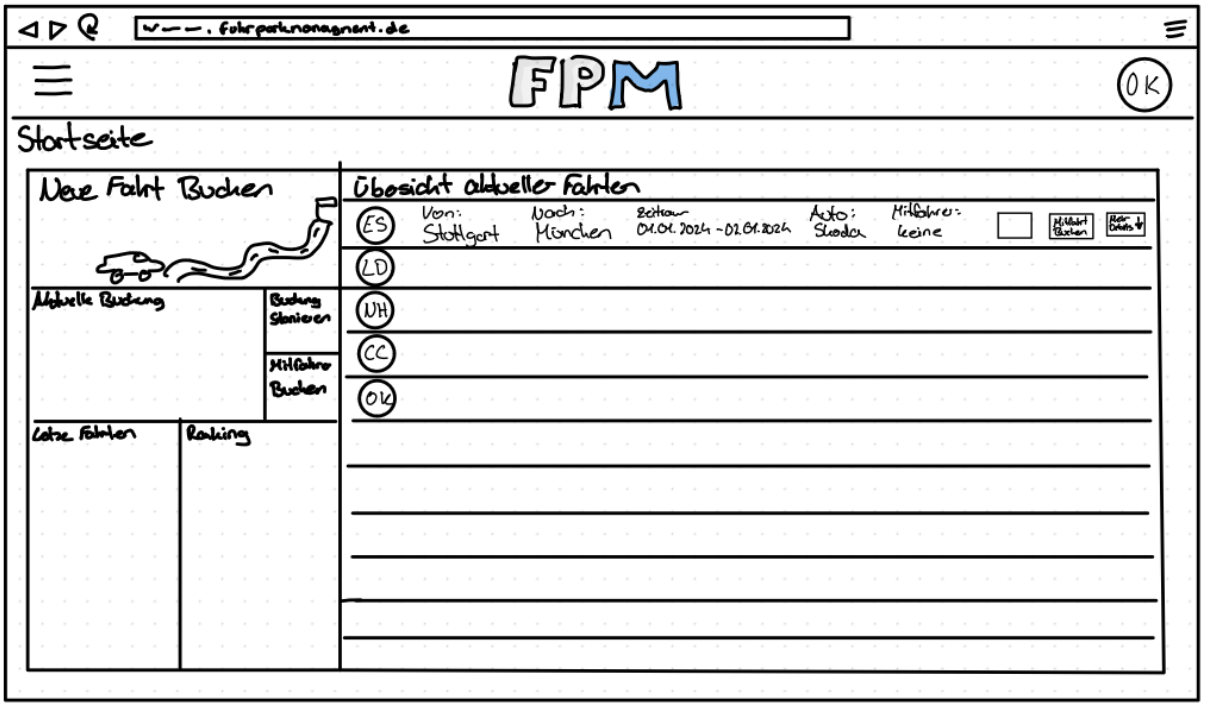
- **Responsive Webanwendung*:*** Die Benutzeroberfläche sollte so konzipiert sein, dass verschiedene Geräte und Bildschirmgrößen optimal funktionieren. So ist es auch möglich, dass man diese auch von unterwegs nutzen kann.
- **Zentrale Plattform*:*** Das System schafft eine zentrale Plattform zur Buchung und Verwaltung von Firmenfahrzeugen. Damit sind auch doppelte Buchungen und sonstige Unstimmigkeiten nicht mehr möglich. Alle Daten sind an einem logischen Ort und nicht mehr verteilt.
- **Statistikerfassung*:*** Das System erfasst statistische Daten über die Nutzung der Fahrzeuge, um unter anderem die Ressourcenplanung und -verwaltung zu verbessern.

- **Erweiterbarkeit*:*** Das System sollte mit dem Aspekt der Erweiterbarkeit entwickelt werden, sodass es auch nach Abschluss des Projekts zu erweiterten Anforderungen noch von einer externen Person weiterentwickelt werden könnte.
- **Container-Software-Kompatibilität*:*** Das System kann mittels Container-Software wie Docker virtualisiert werden, um eine effizientere Ausführung (und ggfs. Skalierbarkeit) zu gewährleisten.

# UI Entwürfe

Um eine Idee zu bekommen, wie die Anwendung aussehen könnte, und worauf wir achten müssen, haben wir eine einfach gehaltene UI Prototyp Skizze erstellt.



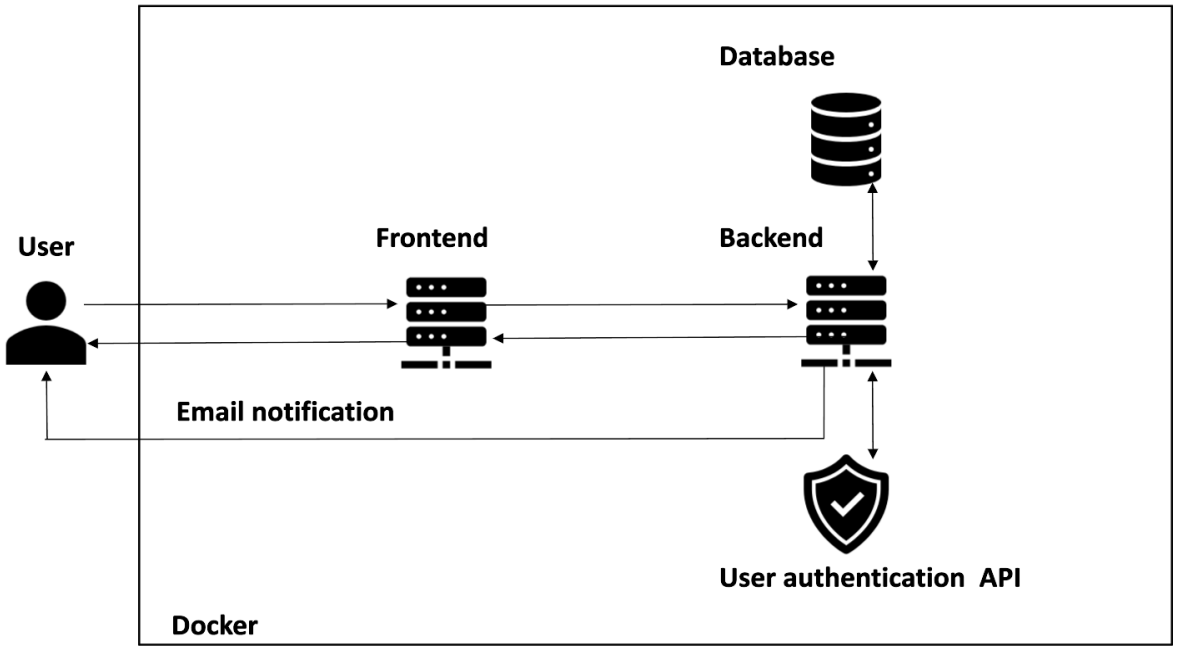



# Architektur / Technologie

Konkrete Vorgaben, welche Technologien wir zu verwenden haben, wurden von der Firma doubleSlash nicht gestellt, jedoch sind Spring Boot und Angular erwünscht.
Die anderen Architektur-/Technologieentscheidungen, wie z.B. Git, Docker etc. haben wir vor allem an Kriterien wie Einsatzzweck, Praxisbewährung oder auch Erlernbarkeit entschieden. Wir haben uns auch über die Lizenzen der einzelnen Technologien informiert und überprüft, ob diese die kommerzielle Nutzung oder Veränderungen zulassen.

- Spring Boot als Backend Framework
    - Gradle als Build-Management-Automatisierungs-Tool
        - Lizenz: Apache 2.0
        - Incremental Builds, Sehr flexibel, Tasks lassen sich einfach erstellen und einbinden
    - JUnit für Unit-Tests
        - Lizenz: Eclipse Public License 2.0
    - Integrationstests mit Spring Boot Test
    - Lizenz: Apache 2.0
    - Wenig Konfigurationsaufwand, einfache Entwicklung von REST-Anwendungen, Ecosystem mit großer Anzahl von Benutzern, Praxisbewährt, (Erwünscht von doubleSlash)
- Angular als Frontend Framework
    - npm als Abhängigkeits-/Paketmanager
        - Lizenz: Artistic License 2.0
    - Testing-Frameworks: Jasmine & Karma
        - Lizenz: MIT
    - Lizenz: MIT
    - Modularisierung durch Komponente-System, in der Praxis weit verbreitet, (Erwünscht von doubleSlash)
- PostgreSQL als Datenbank-Managementsystem (DBMS)
    - Lizenz: PostgreSQL Licence
    - Open Source, hat sich in der Praxis bewährt (Robust), gute SQL Unterstützung
- Git als Version Control System (VCS)
    - GitHub Repository als Code-Hosting-Plattform
    - GitHub Kanban Board für das Projektmanagement / Aufgabenverteilung
    - Github Actions als CI/CD Pipeline
    - Zentrale Verwaltung des Quellcodes, des Projektmanagements und der CI/CD Pipeline auf einer Plattform sichert eine bestmögliche Integration der einzelnen Dienste untereinander und verringert den Arbeitsaufwand
    - Basisfunktionen sind kostenfrei mit optionalen Abonnement Modellen für erweiterte Features und mehr Performance (z.B. mehr Speicherplatz)
- Docker zur Containerization und Virtualisierung
    - Lizenz: Apache 2.0
    - Industriestandard, gute Integration in z.B. Intellij IDEA und hat außerdem einen eigene Desktopanwendung zur Verwaltung von Docker Containern (Docker Desktop)
- IntelliJ IDEA Ultimate als Entwicklungsumgebung (IDE)

- All-in-One Lösung, beinhaltet z.B. auch Git- und Datenbanken Verwaltungs Funktionalität
- Kostenlose Bildungslizenzen für unter anderem Studierende




Erstes Datenmodell

# Projektmanagement

Gruppenintern haben wir uns dazu entschieden, wöchentlich im Team zu präsentieren, was man erreicht hat (Weekly Standup).
Anschließend werden wir mit dem Kunden den aktuellen Stand durchgehen, um sofort zu sehen, ob die Features, die wir schon implementiert haben, und noch implementieren wollen, der Vorstellung entsprechen.
Unser Backlog haben wir in das Kanban Board von GitHub eingetragen. Hier nimmt sich jeder vor, was er bis zur nächsten Woche geschafft haben will.
Was andere Infos oder Absprachen angeht, haben wir einen Discord Server und eine WhatsApp Gruppe erstellt.

Definition of Done (DoD):

- Präsentation für Team vorbereitet (Jeder präsentiert Dienstagabend vor dem Meeting mit Kunden, was er erreicht hat, und sagt, was wichtig zu wissen fürs Team ist)
- Feature Dokumentiert
- Feature manuell getestet, ggfs. Unit-/Integrationstest

# User Stories & Aufwandsschätzung

Wir haben geschätzt, was wir denken, wie viel Aufwand die einzelnen Backlog Items, in Stunden, sein könnten. Dies ist uns sehr schwer gefallen, da unterschiedliche Teammitglieder unterschiedliche Ideen hatten, aber auch natürlich, da das Thema Inhaltlich und das Schätzen allgemein für uns mit wenig Erfahrung verbunden ist.
15 Stunden pro Woche, pro Person, mal fünf Personen, mal acht Wochen für die Implementierung ergibt ein Zeitkontingent von 600 Stunden.
Hier müssen natürlich Puffer für unvorhergesehene Probleme zugerechnet werden und jedes Feature muss dokumentiert und zur Präsentation aufbereitet werden.

- Als Mitarbeiter möchte ich mich mit meinem Doubleslash Account einloggen können.
  - Login Ansicht erstellen: 8 Stunden
  - Session Handling: 10 Stunden
  - Daten über Kunden-API validieren und Token erhalten: 18 Stunden
  - Kundendaten und Token in der Datenbank und Browser halten: 16 Stunden
  - **=> 52h**
- Als Mitarbeiter möchte ich in einer Kalenderansicht einen Zeitraum für meine Buchung auswählen können.
  - Kalenderansicht im Frontend erstellen, die Buchungen über einen oder mehrere Tage zulässt, und start und ende anwählbar macht: 36 Stunden
  - Daten in der Datenbank zu einer Buchung hinzufügen: 24 Stunden
  - **=> 60h**
- Als Mitarbeiter möchte ich bereits gebuchte Fahrten sehen, um zu sehen, wann kein Auto verfügbar ist.
  - Ansicht erstellen, die existierende Buchungen übersichtlich anzeigt: 30 Stunden
  - Möglichkeit, sich bei einer Fahrt hinzuzubuchen: 18 Stunden
  - Möglichkeit, sich bei einer Fahrt zu entfernen: 18 Stunden
  - **=> 66h**
- Als Mitarbeiter möchte ich Firmenstandorte und selbst gesetzte Adressen als Ziel und Startpunkt verwenden können.
  - Adressbuch einbinden, eingegebene Adressen werden automatisch vervollständigt/vorgeschlagen: 36 Stunden
  - Firmenstandorte hinterlegt und direkt auswählbar: 18 Stunden
  - **=> 54h**
- Als Mitarbeiter möchte ich Buchungen löschen können: **=> 18 Stunden**
- Als Mitarbeiter möchte ich Statistiken angezeigt bekommen.
  - OpenStreetView-Berechnung der Kilometer: 30 Stunden
  - Leaderboard-Design: 30 Stunden
  - **=> 60h**
- Als Mitarbeiter möchte ich im Buchungsprozess alle aktuell freien Fahrzeuge auswählen können **=> 30h**
- Als Mitarbeiter möchte ich automatisch einen Outlook-Termin an die Fahrtteilnehmer versenden können.
  - Versenden von ICS-Datei: 24 Stunden
  - Konfiguration SMTP: 22 Stunden

    - **=> 46h**
- Als Mitarbeiter möchte ich Zwischenstopps zu einer Route hinzufügen können: 30 Stunden
- Als Mitarbeiter möchte ich eine Buchung vorzeitig beenden können: 18 Stunden

Insgesamt sind es ca. 386 Stunden für die Features. Puffer ca. 20%: 100 Stunden. Meetings mit dem Kunden: ca. 40 Stunden. Da wir uns in einem agilen Umfeld bewegen kommen für regelmäßige Überarbeitungen der Anforderungen noch Zeit hinzu: ca. 50h.

| Aufgaben | Wochen | KW40 | KW41 | KW42 | KW43 | KW44 | KW45 | KW46 | KW47 | KW48 | KW49 | KW50 | KW51 | KW52 | KW01 | KW02 | KW02 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Phase 1: Planung** | | | | | | | | | | | | | Weihnachten | | | | Endpräsentation |
| Kundenanforderung | 2 | | | | | | | | | | | | | | | | |
| Verwendung von Technologien | 2 | | | | | | | | | | | | | | | | |
| Zeit und Aufgabenmanagement | 2 | | | | Zwischenpräsentation | | | | | | | | | | | | |
| **Phase 3: Vorbereitung** | | | | | | | | | | | | | | | | | |
| Einrichtung & Einarbeitung der Technologien | 2 | | | | | | | | | | | | | | | | |
| Initialisierung | 2 | | | | | | | | | | | | | | | | |
| **Phase 4: Umsetzung** | | | | | | | | | | | | | | | | | |
| Entwicklung | 8 | | | | | | | | | | | | | | | | |
| Tests | 8 | | | | | | | | | | | | | | | | |
| **Phase 5: Dokumentation** | | | | | | | | | | | | | | | | | |
| Dokumentation fertigstellen | 2 | | | | | | | | | | | | | | | | |

Zeitplan