Projekt „Terminverwaltung“

Inhalt

[2 Einführung 3](#_Toc147854452)

[2.1 Kernfunktion 3](#_Toc147854453)

[3 Funktionen 3](#_Toc147854454)

[3.1 Terminverwaltung: 4](#_Toc147854455)

[3.2 Rabattverwaltung: 4](#_Toc147854456)

[3.3 Benutzer- und Mitarbeiterverwaltung: 4](#_Toc147854457)

[3.4 Benachrichtigungen 4](#_Toc147854458)

[4 Technische Details 4](#_Toc147854459)

[4.1 Technologiestack: 4](#_Toc147854460)

[4.2 Architektur: 4](#_Toc147854461)

[4.3 Datenschutz und Sicherheit: 4](#_Toc147854462)

[4.4 Entwicklungsprozess 4](#_Toc147854463)

[4.5 Fazit 4](#_Toc147854464)

[5 Technische Beschreibung einer Terminverwaltungswebanwendung für ein Dienstleistungs-KMU 4](#_Toc147854465)

[5.1 Einführung 4](#_Toc147854466)

[5.2 Funktionen 4](#_Toc147854467)

[6 Die Webanwendung wird die folgenden Hauptfunktionen bieten: 4](#_Toc147854468)

[6.1 Terminverwaltung: 4](#_Toc147854469)

[6.2 Rabattverwaltung: 4](#_Toc147854470)

[6.3 Benutzer- und Mitarbeiterverwaltung: 4](#_Toc147854471)

[6.4 Benachrichtigungen: 4](#_Toc147854472)

[7 Technische Details 5](#_Toc147854473)

[7.1 Technologiestack: 5](#_Toc147854474)

[7.2 Architektur: 5](#_Toc147854475)

[7.3 Datenschutz und Sicherheit: 5](#_Toc147854476)

[8 Entwicklungsprozess 5](#_Toc147854477)

[9 Fazit 6](#_Toc147854478)

[10 Einführung 6](#_Toc147854479)

[11 Funktionen 6](#_Toc147854480)

[11.1 2.1. Terminverwaltung: 7](#_Toc147854481)

[11.2 2.2. Rabattverwaltung: 7](#_Toc147854482)

[11.3 2.3. Benutzer- und Mitarbeiterverwaltung: 7](#_Toc147854483)

[11.4 2.4. Benachrichtigungen: 7](#_Toc147854484)

[12 3. Technische Details 7](#_Toc147854485)

[12.1 3.1. Technologiestack: 7](#_Toc147854486)

[12.2 3.2. Architektur: 8](#_Toc147854487)

[12.3 3.3. Datenschutz und Sicherheit: 8](#_Toc147854488)

[13 4. Entwicklungsprozess 8](#_Toc147854489)

[13.1 5. Fazit 8](#_Toc147854490)

[14 Hier sind einige User Stories für Administratoren, Kunden und Mitarbeiter für die Terminverwaltungswebanwendung: 9](#_Toc147854491)

[15 Hier sind die User Stories mit den entsprechenden Use Cases, Abgrenzungen, Validierungen und Testfällen für Administratoren, Kunden und Mitarbeiter: 10](#_Toc147854492)

[15.1 Use Case 1.1: 10](#_Toc147854493)

[15.2 Use Case 1.2 11](#_Toc147854494)

[15.3 Use Case 1.3 11](#_Toc147854495)

[15.4 Use Case 1.4 12](#_Toc147854496)

[15.5 Use Case 1.5 12](#_Toc147854497)

[15.6 Use Case 1.6 13](#_Toc147854498)

[15.7 Use Case 1.7 13](#_Toc147854499)

[15.8 Use Case 1.8 15](#_Toc147854500)

[15.9 Use Case 1.9 16](#_Toc147854501)

[-Der Salon-Manager meldet sich in der Beauty-Salon-Webapplikation an. 18](#_Toc147854502)

[16 Kernfeature: Terminverwaltung mit Überprüfung nicht vollständig vergebener Termine und Rabattfunktion 23](#_Toc147854503)

[17 Überprüfung nicht vollständig vergebener Termine: 23](#_Toc147854504)

[17.1 Funktionalität: 23](#_Toc147854505)

[17.2 Möglichkeit, Termine mit Rabatten zu versehen: 24](#_Toc147854506)

[17.2.1 Funktionalität: 24](#_Toc147854507)

[18 Masken und Controls 25](#_Toc147854508)

[18.1 Anmelde-Maske 25](#_Toc147854509)

[18.2 Administrations-Dashboard 25](#_Toc147854510)

[18.3 Kunden-Dashboard 25](#_Toc147854511)

[18.4 Mitarbeiter-Dashboard 25](#_Toc147854512)

[19 Detailbeschreibungen für die Entwicklung: 25](#_Toc147854513)

[19.1 Anmelde-Maske: 25](#_Toc147854514)

[19.2 Administrations-Dashboard: 26](#_Toc147854515)

[19.3 Kunden-Dashboard: 26](#_Toc147854516)

[19.4 Mitarbeiter-Dashboard: 26](#_Toc147854517)

[20 Todos für den Datenbankentwickler im Zusammenhang mit der Erstellung der MongoDB-Datenbank für die Terminverwaltungswebanwendung: 27](#_Toc147854518)

[20.1 Todos für den Datenbankentwickler: 27](#_Toc147854519)

[21 Einige Todos für den Softwareentwickler im Zusammenhang mit der Entwicklung der Terminverwaltungswebanwendung in Java: 31](#_Toc147854520)

[21.1 Todos für den Softwareentwickler: 31](#_Toc147854521)

[21.2 Java-Klassen und Funktionen: 32](#_Toc147854522)

[21.3 1. Datenmodelle: 32](#_Toc147854523)

[21.4 2. Repository-Klassen: 32](#_Toc147854524)

[21.5 3. Controller-Klassen: 33](#_Toc147854525)

[21.6 4. Service-Klassen: 33](#_Toc147854526)

[21.7 5. Sicherheits-Klassen: 33](#_Toc147854527)

[22 Hier sind Epics (große, übergreifende Aufgaben) für die zuvor erstellten User Stories für die Terminverwaltungswebanwendung: 33](#_Toc147854528)

[22.1 Administratoren: 33](#_Toc147854529)

[22.2 Kunden: 34](#_Toc147854530)

[22.3 Mitarbeiter: 34](#_Toc147854531)

[23 Entwicklung der Terminverwaltungswebanwendung 34](#_Toc147854532)

[24 Terminplan 36](#_Toc147854533)

# Einführung

Die vorgeschlagene Webanwendung ist eine Terminverwaltungssoftware, die speziell für ein Dienstleistungs-KMU entwickelt wird. Diese Anwendung ermöglicht es dem Unternehmen, Termine sowohl für Mitarbeiter als auch für Kunden effizient zu verwalten. Darüber hinaus bietet sie die Möglichkeit, Rabatte auf Termine zu gewähren, die zu Zeiten mit niedriger Kundenfrequenz gebucht werden.

## Kernfunktion

Die Kernfunktion der Terminverwaltung ermöglicht es, die Buchung von Terminen für Kunden und Mitarbeitern effizient zu organisieren. Die Überprüfung nicht vollständig vergebener Termine bietet Transparenz, während die Rabattfunktion das Unternehmen dabei unterstützt, seine Auslastung zu optimieren und Kunden zu motivieren, Termine in Zeiten mit niedriger Kundenfrequenz zu buchen. Diese Kombination von Funktionen verbessert die Effizienz und Wirtschaftlichkeit der Terminverwaltung erheblich.

# Funktionen

Die Webanwendung wird die folgenden Hauptfunktionen bieten:

## Terminverwaltung:

* Mitarbeiter und Kunden können sich in die Anwendung einloggen, um Termine zu erstellen, zu bearbeiten oder zu löschen.
* Die Anwendung wird einen Kalender anzeigen, der die Verfügbarkeit der Mitarbeiter anzeigt.
* Kunden können Termine aus den verfügbaren Zeiten auswählen und buchen.

## Rabattverwaltung:

* Die Anwendung ermöglicht es dem Unternehmen, Zeiten mit niedriger Kundenfrequenz zu definieren, z.B. außerhalb der Stoßzeiten oder an bestimmten Wochentagen.
* Rabatte können auf Termine in diesen Zeiten angewendet werden, um Kunden anzulocken.

## Benutzer- und Mitarbeiterverwaltung:

* Administratoren können Mitarbeiter hinzufügen, bearbeiten oder löschen.
* Mitarbeiter haben eigene Konten, auf denen sie ihre Verfügbarkeit und Termine verwalten können.

## Benachrichtigungen

- Die Anwendung kann automatische Benachrichtigungen an Kunden senden, um sie an ihre bevorstehenden Termine zu erinnern.

- Benachrichtigungen können auch für Terminänderungen oder -stornierungen verschickt werden.

# Technische Details

## Technologiestack:

Die Webanwendung wird unter Verwendung moderner Technologien entwickelt:

- Programmiersprachen: HTML, CSS, JavaScript

- Front-End-Framework: Angular oder React

- Back-End-Technologie: Node.js oder Python

- Datenbank: MongoDB oder MySQL

- Server: AWS, Azure oder Google Cloud

- Sicherheitsprotokolle: OAuth 2.0 für Authentifizierung, HTTPS für Datenverschlüsselung

- Versionierung: Git

## Architektur:

Die Anwendung wird in einer clientseitigen (Single-Page-Application) oder serverseitigen (traditionellen MVC) Architektur entwickelt, abhängig von den Anforderungen und Präferenzen des Unternehmens. Die Kommunikation zwischen Front-End und Back-End erfolgt über APIs.

## Datenschutz und Sicherheit:

Die Anwendung wird strenge Datenschutzrichtlinien und Sicherheitsmaßnahmen implementieren, um die persönlichen Daten der Kunden und Mitarbeiter zu schützen. Dies umfasst die Verwendung von sicheren Authentifizierungsverfahren und Verschlüsselung der Datenübertragung.

## Entwicklungsprozess

Die Entwicklung erfolgt in mehreren Phasen:

- Anforderungsanalyse: Festlegung der genauen Anforderungen in enger Zusammenarbeit mit dem Kunden.

- Front-End- und Back-End-Entwicklung: Implementierung der Funktionalitäten und Integration von Datenbanken.

- Testing und Qualitätssicherung: Umfangreiche Tests, um Fehler zu identifizieren und zu beheben.

- Bereitstellung und Wartung: Hochladen der Anwendung auf den Hosting-Server und laufende Wartung und Aktualisierung.

## Fazit

Die vorgeschlagene Webanwendung zur Terminverwaltung bietet dem Dienstleistungs-KMU eine effiziente Möglichkeit, Termine zu organisieren und Rabatte in Zeiten mit niedriger Kundenfrequenz anzubieten. Durch die Nutzung moderner Technologien und bewährter Sicherheitsmaßnahmen wird die Anwendung den Anforderungen des Unternehmens gerecht und ermöglicht eine reibungslose Verwaltung von Terminen und Kundenbeziehungen.

# Technische Beschreibung einer Terminverwaltungswebanwendung für ein Dienstleistungs-KMU

## Einführung

Die vorgeschlagene Webanwendung ist eine Terminverwaltungssoftware, die speziell für ein Dienstleistungs-KMU entwickelt wird. Diese Anwendung ermöglicht es dem Unternehmen, Termine sowohl für Mitarbeiter als auch für Kunden effizient zu verwalten. Darüber hinaus bietet sie die Möglichkeit, Rabatte auf Termine zu gewähren, die zu Zeiten mit niedriger Kundenfrequenz gebucht werden.

## Funktionen

# Die Webanwendung wird die folgenden Hauptfunktionen bieten:

## Terminverwaltung:

- Mitarbeiter und Kunden können sich in die Anwendung einloggen, um Termine zu erstellen, zu bearbeiten oder zu löschen.

- Die Anwendung wird einen Kalender anzeigen, der die Verfügbarkeit der Mitarbeiter anzeigt.

- Kunden können Termine aus den verfügbaren Zeiten auswählen und buchen.

## Rabattverwaltung:

- Die Anwendung ermöglicht es dem Unternehmen, Zeiten mit niedriger Kundenfrequenz zu definieren, z.B. außerhalb der Stoßzeiten oder an bestimmten Wochentagen.

- Rabatte können auf Termine in diesen Zeiten angewendet werden, um Kunden anzulocken.

## Benutzer- und Mitarbeiterverwaltung:

- Administratoren können Mitarbeiter hinzufügen, bearbeiten oder löschen.

- Mitarbeiter haben eigene Konten, auf denen sie ihre Verfügbarkeit und Termine verwalten können.

## Benachrichtigungen:

- Die Anwendung kann automatische Benachrichtigungen an Kunden senden, um sie an ihre bevorstehenden Termine zu erinnern.

- Benachrichtigungen können auch für Terminänderungen oder -stornierungen verschickt werden.

# Technische Details

## Technologiestack:

Die Webanwendung wird unter Verwendung moderner Technologien entwickelt:

- Programmiersprachen: HTML, CSS, JavaScript

- Front-End-Framework: Angular oder React

- Back-End-Technologie: Node.js oder Python

- Datenbank: MongoDB oder MySQL

- Server: AWS, Azure oder Google Cloud

- Sicherheitsprotokolle: OAuth 2.0 für Authentifizierung, HTTPS für Datenverschlüsselung

- Versionierung: Git

## Architektur:

Die Anwendung wird in einer clientseitigen (Single-Page-Application) oder serverseitigen (traditionellen MVC) Architektur entwickelt, abhängig von den Anforderungen und Präferenzen des Unternehmens. Die Kommunikation zwischen Front-End und Back-End erfolgt über APIs.

## Datenschutz und Sicherheit:

Die Anwendung wird strenge Datenschutzrichtlinien und Sicherheitsmaßnahmen implementieren, um die persönlichen Daten der Kunden und Mitarbeiter zu schützen. Dies umfasst die Verwendung von sicheren Authentifizierungsverfahren und Verschlüsselung der Datenübertragung.

# Entwicklungsprozess

Die Entwicklung erfolgt in mehreren Phasen:

- Anforderungsanalyse: Festlegung der genauen Anforderungen in enger Zusammenarbeit mit dem Kunden.

- Design und UI/UX-Entwicklung: Erstellung des Benutzeroberflächen-Designs für eine benutzerfreundliche Erfahrung.

- Front-End- und Back-End-Entwicklung: Implementierung der Funktionalitäten und Integration von Datenbanken.

- Testing und Qualitätssicherung: Umfangreiche Tests, um Fehler zu identifizieren und zu beheben.

- Bereitstellung und Wartung: Hochladen der Anwendung auf den Hosting-Server und laufende Wartung und Aktualisierung.

# Fazit

Die vorgeschlagene Webanwendung zur Terminverwaltung bietet dem Dienstleistungs-KMU eine effiziente Möglichkeit, Termine zu organisieren und Rabatte in Zeiten mit niedriger Kundenfrequenz anzubieten. Durch die Nutzung moderner Technologien und bewährter Sicherheitsmaßnahmen wird die Anwendung den Anforderungen des Unternehmens gerecht und ermöglicht eine reibungslose Verwaltung von Terminen und Kundenbeziehungen.

Technische Beschreibung einer Terminverwaltungswebanwendung für ein Dienstleistungs-KMU mit Java und MongoDB

# Einführung

Die vorgeschlagene Webanwendung ist eine Terminverwaltungssoftware, die speziell für ein Dienstleistungs-KMU entwickelt wird. Diese Anwendung ermöglicht es dem Unternehmen, Termine sowohl für Mitarbeiter als auch für Kunden effizient zu verwalten. Darüber hinaus bietet sie die Möglichkeit, Rabatte auf Termine zu gewähren, die zu Zeiten mit niedriger Kundenfrequenz gebucht werden.

# Funktionen

Die Webanwendung wird die folgenden Hauptfunktionen bieten:

## 2.1. Terminverwaltung:

- Mitarbeiter und Kunden können sich in die Anwendung einloggen, um Termine zu erstellen, zu bearbeiten oder zu löschen.

- Die Anwendung wird einen Kalender anzeigen, der die Verfügbarkeit der Mitarbeiter anzeigt.

- Kunden können Termine aus den verfügbaren Zeiten auswählen und buchen.

## 2.2. Rabattverwaltung:

- Die Anwendung ermöglicht es dem Unternehmen, Zeiten mit niedriger Kundenfrequenz zu definieren, z.B. außerhalb der Stoßzeiten oder an bestimmten Wochentagen.

- Rabatte können auf Termine in diesen Zeiten angewendet werden, um Kunden anzulocken.

## 2.3. Benutzer- und Mitarbeiterverwaltung:

- Administratoren können Mitarbeiter hinzufügen, bearbeiten oder löschen.

- Mitarbeiter haben eigene Konten, auf denen sie ihre Verfügbarkeit und Termine verwalten können.

## 2.4. Benachrichtigungen:

- Die Anwendung kann automatische Benachrichtigungen an Kunden senden, um sie an ihre bevorstehenden Termine zu erinnern.

- Benachrichtigungen können auch für Terminänderungen oder -stornierungen verschickt werden.

# 3. Technische Details

## 3.1. Technologiestack:

Die Webanwendung wird unter Verwendung moderner Technologien entwickelt:

- Programmiersprache: Java

- Web-Framework: Spring Boot

- Front-End-Framework: Angular oder React

- Datenbank: MongoDB

- Server: AWS, Azure oder Google Cloud

- Sicherheitsprotokolle: OAuth 2.0 für Authentifizierung, HTTPS für Datenverschlüsselung

- Versionierung: Git

## 3.2. Architektur:

Die Anwendung wird in einer clientseitigen (Single-Page-Application) oder serverseitigen (traditionellen MVC) Architektur entwickelt, abhängig von den Anforderungen und Präferenzen des Unternehmens. Die Kommunikation zwischen Front-End und Back-End erfolgt über APIs.

## 3.3. Datenschutz und Sicherheit:

Die Anwendung wird strenge Datenschutzrichtlinien und Sicherheitsmaßnahmen implementieren, um die persönlichen Daten der Kunden und Mitarbeiter zu schützen. Dies umfasst die Verwendung von sicheren Authentifizierungsverfahren und Verschlüsselung der Datenübertragung.

# 4. Entwicklungsprozess

Die Entwicklung erfolgt in mehreren Phasen:

- Anforderungsanalyse: Festlegung der genauen Anforderungen in enger Zusammenarbeit mit dem Kunden.

- Design und UI/UX-Entwicklung: Erstellung des Benutzeroberflächen-Designs für eine benutzerfreundliche Erfahrung.

- Front-End- und Back-End-Entwicklung: Implementierung der Funktionalitäten und Integration von MongoDB für die Datenverwaltung.

- Testing und Qualitätssicherung: Umfangreiche Tests, um Fehler zu identifizieren und zu beheben.

- Bereitstellung und Wartung: Hochladen der Anwendung auf den Hosting-Server und laufende Wartung und Aktualisierung.

## 5. Fazit

Die vorgeschlagene Webanwendung zur Terminverwaltung bietet dem Dienstleistungs-KMU eine effiziente Möglichkeit, Termine zu organisieren und Rabatte in Zeiten mit niedriger Kundenfrequenz anzubieten. Durch die Nutzung moderner Technologien und bewährter Sicherheitsmaßnahmen wird die Anwendung den Anforderungen des Unternehmens gerecht und ermöglicht eine reibungslose Verwaltung von Terminen und Kundenbeziehungen.

# User Stories für Administratoren, Kunden und Mitarbeiter für die Terminverwaltungswebanwendung:

**Administratoren**:

1. Als Administrator möchte ich mich in das System einloggen können, um Zugriff auf alle administrativen Funktionen zu haben.

2. Als Administrator möchte ich neue Mitarbeiter hinzufügen können, indem ich ihre persönlichen Informationen und Arbeitszeiten festlege.

3. Als Administrator möchte ich Mitarbeiter bearbeiten können, um ihre Daten oder Arbeitszeiten zu aktualisieren.

4. Als Administrator möchte ich Mitarbeiter deaktivieren oder löschen können, wenn sie das Unternehmen verlassen.

5. Als Administrator möchte ich Rabattzeiten definieren können, indem ich festlege, welche Zeiten als Stoßzeiten gelten und Rabatte gewährt werden.

6. Als Administrator möchte ich die Übersicht über alle Kunden und ihre Buchungen haben.

7. Als Administrator möchte ich Berichte über die Auslastung der Mitarbeiter und den Umsatz generieren können.

8. Als Administrator möchte ich E-Mail-Benachrichtigungen für wichtige Aktivitäten im System erhalten, wie z.B. neue Buchungen oder Terminänderungen.

**Kunden**:

1. Als Kunde möchte ich mich in das System einloggen können, um meine Buchungen anzuzeigen und neue Termine zu buchen.

2. Als Kunde möchte ich die Verfügbarkeit der Mitarbeiter sehen und Termine in freien Zeiten auswählen können.

3. Als Kunde möchte ich eine Bestätigungs-E-Mail erhalten, sobald ich einen Termin gebucht habe.

4. Als Kunde möchte ich Erinnerungsbenachrichtigungen per E-Mail oder SMS für meine bevorstehenden Termine erhalten.

5. Als Kunde möchte ich Termine bearbeiten oder stornieren können, wenn sich meine Pläne ändern.

6. Als Kunde möchte ich Informationen über Rabatte für Termine außerhalb der Stoßzeiten sehen und sie bei der Buchung nutzen können.

**Mitarbeiter**:

1. Als Mitarbeiter möchte ich mich in das System einloggen können, um meine Verfügbarkeit und Buchungen zu verwalten.

2. Als Mitarbeiter möchte ich meinen Arbeitszeitplan einsehen und aktualisieren können.

3. Als Mitarbeiter möchte ich neue Buchungen akzeptieren oder ablehnen können, um meine Verfügbarkeit zu verwalten.

4. Als Mitarbeiter möchte ich eine Liste meiner bevorstehenden Termine sehen und Benachrichtigungen für neue Buchungen erhalten.

5. Als Mitarbeiter möchte ich die Möglichkeit haben, Notizen zu meinen Terminen hinzuzufügen, um wichtige Informationen zu speichern.

6. Als Mitarbeiter möchte ich eine Übersicht über die von mir erzielten Einnahmen und Rabatte haben.

Diese User Stories dienen als Ausgangspunkt für die Entwicklung und werden im Laufe des Projekts möglicherweise weiter verfeinert und ergänzt. Sie helfen dabei, die Anforderungen und Erwartungen der verschiedenen Benutzergruppen klar zu definieren.

# Use Cases:

## Use Case 1.1:

|  |  |
| --- | --- |
| **Use Case 1.1: Terminvereinbarung über die Webapplikation** | |
| Name | Terminvereinbarung über Webapplikation |
| Kurzbeschreibung | Der Kunde kann einen Termin über die Webapplikation vereinbaren. |
| Akteure | Kunde, Beauty-Salon, Mitarbeiter |
| Auslöser | Der Kunde möchte einen Termin vereinbaren. |
| Ergebnis(se) | Terminbestätigung wird an den Kunden gesendet, der Termin wird im System gespeichert. |
| Vorbedingungen | Der Kunde hat sich in der Salon-Webapplikation angemeldet. |
| Eingehende Daten | Gewünschtes Datum und Uhrzeit für den Termin. |
| Ablaufbeschreibung | 1. Der Kunde öffnet die Salon-Webapplikation.  2. Der Kunde wählt das gewünschte Datum und die Uhrzeit aus.  3. Der Kunde bestätigt die Terminvereinbarung.  4. Das System sendet eine Bestätigung an den Kunden |
| Fehlerverhalten | Wenn das Datum oder die Uhrzeit nicht verfügbar ist, wird dem Kunden eine Fehlermeldung angezeigt. |
| Variationen | Keine vorhanden |

## Use Case 1.5

|  |  |
| --- | --- |
| **Use Case 1.5 Zahlungsmöglichkeiten für Kunden auswählen** | |
| Name | Zahlungsmöglichkeiten für Kunden auswählen. |
| Kurzbeschreibung | Der Kunde kann die gewünschte Zahlungsmethode (PayPal, Kreditkarte oder von Ort) auswählen. |
| Akteure | Kunde, Zahlungsabwicklungs-Schnittstelle (z.B. PayPal API, Kreditkarten-Abrechnungssystem) |
| Auslöser | Der Kunde möchte für die erhaltenen Dienstleistungen bezahlen. |
| Ergebnis(se) | Der Zahlungsbetrag wird erfolgreich vom Kundenkonto abgebucht und auf das Konto des Beauty-Salons überwiesen. |
| Vorbedingungen | 1.Der Kunde hat die Dienstleistungen im Beauty-Salon in Anspruch genommen. |
| Eingehende Daten | Der Gesamtbetrag für die erbrachten Dienstleistungen. |
| Ablaufbeschreibung | 1.Der Kunde wählt die Option "Bezahlen" in der Salon-Web oder auf der Website aus.  2.Das System überprüft die eingegebenen Informationen und führt die Zahlungstransaktion durch.  3.Der Kunde erhält eine Bestätigung über die erfolgreiche Zahlung und eine Quittung per E-Mail oder in der Webapplikation. |
| Fehlerverhalten | Bei technischen Problemen, wie z.B. Serverausfällen. |
| Variationen | Keine vorhanden |

## Use Case 1.6

|  |  |
| --- | --- |
| **Use Case 1.6: Teamfotos anzeigen über die Webapplikation** | |
| Name | Teamfotos anzeigen über die Webapplikation |
| Kurzbeschreibung | Der Kunde kann Fotos des Friseurteams im Salon über die Salon-Webapplikation anzeigen. |
| Akteure | Kunde, Salon-Webapplikation |
| Auslöser | Der Kunde möchte Fotos des Friseurteams sehen, um das Salonpersonal kennenzulernen. |
| Ergebnis(se) | Der Kunde kann Fotos des Friseurteams anzeigen und das Team kennenlernen. |
| Vorbedingungen | Der Kunde hat die Salon-Webapplikation installiert und ist angemeldet. |
| Eingehende Daten | Keine spezifischen Daten erforderlich. |
| Ablaufbeschreibung | 1.Der Kunde öffnet die Salon-Webapplikation.  2.Der Kunde navigiert zur Option "Unser Team".  3.Das System zeigt eine Galerie von Fotos des Friseurteams im Salon an.  4.Der Kunde kann jedes Teammitglied anklicken, um mehr über ihre Fähigkeiten, Erfahrungen und Spezialgebiete zu erfahren. |
| Fehlerverhalten | Wenn keine Teamfotos verfügbar sind. |
| Variationen | Kurze Vorstellungsvideos der Teammitglieder könnten ebenfalls verfügbar sein, um eine persönliche Verbindung zu schaffen. |

## Use Case 1.7

|  |  |
| --- | --- |
| **Use Case 1.7: Über Uns und Kontaktinformationen anzeigen über die Webapplikation** | |
| Name | Über Uns und Kontaktinformationen anzeigen über die Webapplikation |
| Kurzbeschreibung | Der Kunde kann sich über das Salon-Team informieren und Kontaktinformationen anzeigen, einschließlich Vorstellungsvideos oder Fotos mit kurzen Texten von jedem Mitarbeiter. |
| Akteure | Kunde, Salon-Webapplikation, Mitarbeiter |
| Auslöser | Der Kunde möchte Informationen über das Salon-Team erfahren oder Kontakt mit einem bestimmten Mitarbeiter aufnehmen. |
| Ergebnis(se) | Der Kunde kann sich über das Salon-Team informieren, Mitarbeiterprofile anzeigen und Kontaktinformationen anzeigen. |
| Vorbedingungen | Der Kunde hat die Salon-Webapplikation installiert und ist angemeldet. |
| Eingehende Daten | Keine spezifischen Daten erforderlich. |
| Ablaufbeschreibung | 1.Der Kunde öffnet die Salon-Webapplikation.  2.Der Kunde navigiert zur Option "Über Uns" oder "Unser Team" im Hauptmenü.  3.Das System zeigt eine Liste der Mitarbeiter mit ihren Profilbildern an.  4.Der Kunde kann jedes Mitarbeiterprofil anklicken, um ein Vorstellungsvideo oder ein Foto mit einem kurzen Text über den Mitarbeiter anzuzeigen.  5.Falls gewünscht, kann der Kunde über die Webapplikation direkt Kontakt mit dem Mitarbeiter aufnehmen, z. B. durch einen Anruf, eine Nachricht oder eine Terminvereinbarung. |
| Fehlerverhalten | Wenn keine Mitarbeiterprofile verfügbar sind. |
| Variationen | 1.Mitarbeiter könnten kurze Vorstellungsvideos hochladen, um sich persönlich vorzustellen.  2.Die Option für den Kunden, direkt über die Webapplikation einen Termin mit einem bestimmten Mitarbeiter zu vereinbaren. |

## Use Case 1.8

|  |  |
| --- | --- |
| **Use Case 1.8 Adresse des Beauty-Salons anzeigen über Google Maps** | |
| Name | Adresse des Beauty-Salons anzeigen über Google Maps |
| Kurzbeschreibung | Der Kunde kann die Adresse des Beauty-Salons über Google Maps anzeigen lassen. |
| Akteure | Kunde, Google Maps API (für die Integration der Kartendaten) |
| Auslöser | Der Kunde möchte den Beauty-Salon besuchen und benötigt die genaue Adresse und Wegbeschreibung. |
| Ergebnis(se) | 1. Der Kunde sieht den Standort des Beauty-Salons auf einer interaktiven Karte.  2.Der Kunde kann die Wegbeschreibung zum Beauty-Salon anzeigen lassen. |
| Vorbedingungen | 1.Der Kunde hat eine Internetverbindung.  2.Der Beauty-Salon ist auf Google Maps registriert und seine Adresse ist korrekt hinterlegt |
| Eingehende Daten | Keine spezifischen Daten, außer der Anfrage des Kunden, die Adresse des Beauty-Salons anzuzeigen. |
| Ablaufbeschreibung | 1.Der Kunde öffnet die Salon-Website oder die Salon-Webapplikation.  2.Auf der Website oder in der Webapplikation gibt es eine Schaltfläche oder einen Link mit der Bezeichnung "Standort anzeigen", der mit Google Maps verbunden ist.  4.Die Website oder Webapplikation integriert Google Maps und zeigt den genauen Standort des Beauty-Salons auf einer Karte an. |
| Fehlerverhalten | 1.Wenn die Adresse des Beauty-Salons nicht korrekt auf Google Maps hinterlegt ist, wird dem Kunden eine Fehlermeldung angezeigt.  2.Bei technischen Problemen, wie z.B. dem Ausfall der Google Maps API, wird dem Kunden eine Fehlermeldung angezeigt, und es wird darauf hingewiesen, es später erneut zu versuchen. |
| Variationen | Keine vorhanden |

## Use Case 1.9

|  |  |
| --- | --- |
| **Use Case 1.9 Administrator-Anmeldung** | |
| Name | Administrator-Anmeldung und Salonverwaltung |
| Kurzbeschreibung | Administrator-Anmeldung für die Webapplikation eines Beauty-Salons. |
| Akteure | Administrator des Beauty-Salons. |
| Auslöser | Der Administrator möchte auf die Administrationsfunktionen der Webapplikation zuzugreifen. |
| Ergebnis(se) | Der Administrator hat erfolgreich Zugriff auf die Administrationsfunktionen.  Fehlermeldung, wenn die Anmeldung fehlschlägt. |
| Vorbedingungen | Die Webapplikation des Beauty-Salons ist korrekt eingerichtet und läuft.  Der Administrator verfügt über gültige Anmeldeinformationen (Benutzername und Passwort. |
| Eingehende Daten | Benutzername: Die eindeutige Identifikation des Administrators.  Passwort: Das geheime Kennwort des Administrators. |
| Ablaufbeschreibung | Administrator öffnet die Startseite der Webapplikation im Browser.    Administrator wählt die Anmeldung für Administratoren aus.    Administrator gibt Benutzername und Passwort ein.    Administrator klickt auf "Anmelden".    Webapplikation prüft Anmeldeinformationen.    Bei korrekten Daten: Administrator gelangt zur Administrations-Oberfläche.    Administrator nutzt Administrationsfunktionen.    Bei falschen Daten: Fehlermeldung, erneute Eingabe.    Administrator kann sich abmelden, um die Sitzung zu beenden.    Ende, wenn Administrator Aufgaben abgeschlossen hat oder die Webapplikation schließt. |

|  |  |
| --- | --- |
| **Use Case 1.10 Mitarbeiterverwaltung** | |
| Name | Mitarbeiterverwaltung |
| Kurzbeschreibung | Dieser Use-Case ermöglicht dem Salon-Manager, Mitarbeiterinformationen in der Beauty-Salon-Webapplikation zu verwalten . |
| Akteure | Salon-Manager |
| Auslöser | Der Salon-Manager möchte neue Mitarbeiter hinzufügen, bestehende Mitarbeiter aktualisieren oder löschen oder allgemeine Informationen über Mitarbeiter abrufen. |
| Ergebnis(se) | Neue Mitarbeiter können hinzugefügt werden.  Informationen über bestehende Mitarbeiter können aktualisiert werden.  Mitarbeiter können aus dem System gelöscht werden.  Informationen über Mitarbeiter können abgerufen werden. |
| Vorbedingungen | Der Salon-Manager muss in der Beauty-Salon-Webapplikation angemeldet sein |
| Eingehende Daten | -Name des Mitarbeiters  -Kontaktinformationen des Mitarbeiters (Telefonnummer, E-Mail-Adresse)  -Position oder Jobtitel des Mitarbeiters  -Arbeitszeiten des Mitarbeiters  -Gehaltsinformationen (optional)  -Weitere relevante Informationen (z. B. Qualifikationen, Zertifikate) |
| Ablaufbeschreibung | -Der Salon-Manager meldet sich in der Beauty-Salon-Webapplikation an. -Der Manager navigiert zur Mitarbeiterverwaltungsoberfläche.  -Der Manager wählt die gewünschte Aktion aus: "Mitarbeiter hinzufügen", "Mitarbeiter aktualisieren" oder "Mitarbeiter löschen".  -Wenn der Manager einen neuen Mitarbeiter hinzufügen möchte, gibt er die erforderlichen Informationen ein und bestätigt die Eingabe.  -Wenn der Manager Informationen eines bestehenden Mitarbeiters aktualisieren möchte, wählt er den Mitarbeiter aus, ändert die relevanten Daten und speichert die Aktualisierung.  -Wenn der Manager einen Mitarbeiter löschen möchte, wählt er den Mitarbeiter aus und bestätigt die Löschung.  Der Manager kann auch Informationen über Mitarbeiter abrufen, indem er den entsprechenden Mitarbeiter auswählt und die relevanten Informationen angezeigt werden. |
| **Abgrenzungen:** | Der Administrator hat bereits ein Konto in der Anwendung |
| **Validierungen:** | - Der Administrator gibt gültige Anmeldeinformationen (Benutzername und Passwort) ein.  - Der Administrator füllt die erforderlichen Mitarbeiterinformationen aus. |

|  |  |
| --- | --- |
| **Use Case 1.11 Kundenanmeldung** | |
| Name | Kundenanmeldung |
| Kurzbeschreibung | Dieser Use-Case beschreibt den Prozess, den ein Kunde durchläuft, um sich in der Beauty-Salon-Webapplikation anzumelden. |
| Akteure | Kunde |
| Auslöser | Der Kunde möchte auf die Funktionen und Dienstleistungen der Beauty-Salon-Webapplikation zugreifen. |
| Ergebnis(se) | -Der Kunde ist erfolgreich in der Beauty-Salon-Webapplikation angemeldet.  -Der Kunde kann auf die angebotenen Dienstleistungen, Termine und andere Funktionen der Webapplikation zugreifen. |
| Vorbedingungen | -Der Kunde verfügt über ein Konto in der Beauty-Salon-Webapplikation.  -Der Kunde hat einen Internetzugang und ein geeignetes Gerät. |
| Eingehende Daten | -Benutzername oder E-Mail-Adresse des Kunden  -Passwort des Kunden |
| Ablaufbeschreibung | - Der Kunde öffnet die Beauty-Salon-Webapplikation auf seinem Gerät.  -Der Kunde klickt auf die Schaltfläche "Anmelden" oder "Login".  -Die Webapplikation fordert den Kunden auf, seine Benutzerdaten einzugeben (Benutzername oder E-Mail-Adresse und Passwort).  -Der Kunde gibt seine Benutzerdaten ein.  -Die Webapplikation überprüft die eingegebenen Daten auf ihre Richtigkeit und Validität.  -Bei erfolgreicher Validierung der Daten wird der Kunde in sein Konto eingeloggt, und die Hauptseite der -Webapplikation wird angezeigt.  -Der Kunde kann nun auf die verschiedenen Funktionen und Dienstleistungen der Beauty-Salon-Webapplikation zugreifen. |
| **Abgrenzungen:** | - Der Kunde hat bereits ein Konto in der Anwendung. |
| **Validierungen:** | - Der Kunde gibt gültige Anmeldeinformationen (Benutzername und Passwort) ein. |

|  |  |
| --- | --- |
| **Use Case 1.12 Terminverwaltung** | |
| Name | Terminverwaltung |
| Kurzbeschreibung | Dieser Use-Case beschreibt den Prozess der Verwaltung von Terminen für Kunden in der Beauty-Salon-Webapplikation. |
| Akteure | Kunde, Salonmitarbeiter |
| Auslöser | Ein Kunde möchte einen Termin im Beauty-Salon buchen oder ändern. |
| Ergebnis(se) | -Der Kunde hat erfolgreich einen Termin gebucht oder geändert.  -Der Salonmitarbeiter hat die Terminanfrage bearbeitet und bestätigt oder abgelehnt. |
| Vorbedingungen | Der Kunde ist in der Salon-Webapplikation angemeldet.  Die Webapplikation ist verfügbar und funktionsfähig. |
| Eingehende Daten | -Gewünschtes Datum und Uhrzeit für den Termin.  -Art des gewünschten Services (z.B., Haarschnitt, Maniküre, Gesichtsbehandlung).  -Kundenkontoinformationen (Name, Kontaktinformationen). |
| Ablaufbeschreibung | -Der Kunde meldet sich in der Salon-Webapplikation an.  -Der Kunde navigiert zur Terminverwaltungsoption.  -Der Kunde wählt das gewünschte Datum und die Uhrzeit aus.  -Der Kunde gibt an, welchen Service er wünscht.  -Der Kunde überprüft die Buchungsdetails und bestätigt die Terminanfrage.  -Die Webapplikation übermittelt die Terminanfrage an das Salonpersonal.  -Der Salonmitarbeiter überprüft die Verfügbarkeit und -bestätigt den Termin oder schlägt alternative Zeiten vor.  -Der Kunde erhält eine Benachrichtigung über die Terminbestätigung oder die Alternativvorschläge.  -Bei Terminbestätigung erhält der Kunde eine Erinnerungsnachricht vor dem Termin |
| **Abgrenzungen:** | - Der Kunde ist im System angemeldet und hat bereits Termine. |
| **Validierungen:** | Der Kunde kann bestehende Termine bearbeiten oder stornieren. |

|  |  |
| --- | --- |
| **Use Case 1.13 Mitarbeiteranmeldung** | |
| Name | Anmeldung für Mitarbeiter |
| Kurzbeschreibung | Dieser Use-Case beschreibt den Prozess, bei dem Mitarbeiter sich in die Salon-Webapplikation anmelden. |
| Akteure | Mitarbeiter, Salon-Webapplikation |
| Auslöser | Ein Mitarbeiter möchte auf die Salon-Webapplikation zugreifen. |
| Ergebnis(se) | Der Mitarbeiter hat erfolgreich auf die Salon-Webapplikation zugegriffen.  Die Anmeldedaten des Mitarbeiters wurden überprüft und sind korrekt. |
| Vorbedingungen | Die Salon-Webapplikation ist gestartet und betriebsbereit.  Der Mitarbeiter hat gültige Anmeldedaten (Benutzername und Passwort). |
| Eingehende Daten | -Benutzername  -Passwort |
| Ablaufbeschreibung | -Der Mitarbeiter öffnet die Salon-Webapplikation auf seinem Gerät.  -Auf der Anmeldeseite gibt der Mitarbeiter seinen Benutzernamen und sein Passwort ein.  -Die Webapplikation überprüft die eingegebenen Anmeldedaten auf Korrektheit.  -Wenn die Anmeldedaten korrekt sind, wird der Mitarbeiter zur Hauptansicht der Webapplikation weitergeleitet.  -Falls die Anmeldedaten falsch sind, wird dem Mitarbeiter eine Fehlermeldung angezeigt, und er kann erneut versuchen, sich anzumelden.  -Der Mitarbeiter hat nun Zugriff auf die Funktionen und Informationen, die seinen Berechtigungen entsprechen. |
| **Abgrenzungen:** | - Der Mitarbeiter hat bereits ein Konto in der Anwendung. |
| **Validierungen:** | - Der Mitarbeiter gibt gültige Anmeldeinformationen (Benutzername und Passwort) ein. |

|  |  |
| --- | --- |
| **Use Case 1.14 Verfügbarkeitsverwaltung** | |
| Name | Verfügbarkeitsverwaltung für den Beauty-Salon |
| Kurzbeschreibung | Die Verfügbarkeitsverwaltung für den Beauty-Salon ist ein Funktionsteil der Salon-Webapplikation, der es ermöglicht, die Termine für Kunden und Mitarbeiter effizient zu planen und zu verwalten. |
| Akteure | Salon-Administrator, Schönheitsspezialist, Kunden |
| Auslöser | -Ein Kunde möchte einen Termin im Beauty-Salon vereinbaren oder ändern.  -Ein Mitarbeiter möchte seine Verfügbarkeit anzeigen oder aktualisieren.  -Der Administrator möchte den Salon-Terminkalender verwalten. |
| Ergebnis(se) | -Effiziente Terminplanung für Kunden und Mitarbeiter.  -Reduzierung von Terminüberschneidungen.  -Verbesserte Kundenzufriedenheit durch einfache Terminbuchung. |
| Vorbedingungen | -Die Salon-Webapplikation ist ordnungsgemäß gestartet und funktionstüchtig.  -Mitarbeiter und Kunden sind im System registriert und angemeldet |
| Eingehende Daten | Kundenanfrage für einen Termin (Datum, Uhrzeit, gewünschte Dienstleistung).  Mitarbeiterverfügbarkeit (Arbeitszeiten und Ausnahmen).  Aktueller Terminplan des Beauty-Salons. |
| Ablaufbeschreibung | -Ein Kunde meldet sich in der Salon-Webapplikation an.  -Der Kunde wählt das gewünschte Datum und die gewünschte Uhrzeit aus und gibt die gewünschte Dienstleistung an.  -Die Anfrage wird an den Administrator oder die relevanten Mitarbeiter weitergeleitet.  -Der Administrator prüft die Verfügbarkeit des ausgewählten Mitarbeiters und bestätigt den Termin oder schlägt alternative Termine vor.  -Der Kunde erhält eine Bestätigung des Termins. |
| **Abgrenzungen:** | - Der Mitarbeiter ist im System angemeldet. |
| **Validierungen:** | - Der Mitarbeiter kann seine Verfügbarkeit aktualisieren. |

# Kernfeature: Terminverwaltung mit Überprüfung nicht vollständig vergebener Termine und Rabattfunktion

Das Kernfeature der Terminverwaltung in der Webanwendung ermöglicht es dem Unternehmen, die Terminplanung für Mitarbeiter und Kunden effizient zu organisieren. Dieses Feature enthält zwei wichtige Funktionalitäten: die Überprüfung nicht vollständig vergebener Termine und die Möglichkeit, Termine mit Rabatten zu versehen.

# Überprüfung nicht vollständig vergebener Termine:

## Funktionalität:

- Das System zeigt eine Übersicht über die Verfügbarkeit der Mitarbeiter in Echtzeit an.

- Nicht vollständig vergebene Termine werden farblich hervorgehoben, um sie leicht erkennbar zu machen.

- Die Anwendung ermöglicht es Kunden, Termine in den verfügbaren Zeiten auszuwählen und zu buchen.

**Details**:

- Die Anwendung zeigt den Zeitplan jedes Mitarbeiters an und markiert die bereits gebuchten Termine.

- Ungebuchte Zeiten oder Zeiten mit niedriger Kundenfrequenz werden zur besseren Planung sichtbar gemacht.

- Kunden können Termine in diesen verfügbaren Zeiten direkt buchen.

**Vorteile**:

- Kunden können schnell und einfach freie Termine finden und buchen.

- Mitarbeiter können ihre verbleibende Verfügbarkeit verfolgen und Termine planen.

- Das Unternehmen kann seine Auslastung maximieren, indem es nicht vollständig vergebene Zeiten identifiziert und proaktiv Aktionen ergreift.

## Möglichkeit, Termine mit Rabatten zu versehen:

### Funktionalität:

- Administratoren haben die Möglichkeit, Zeiten mit niedriger Kundenfrequenz oder Stoßzeiten zu definieren.

- Rabatte können auf Termine in diesen Zeiten angewendet werden, um Kunden anzulocken.

- Kunden sehen die Rabatte, wenn sie Termine in diesen Zeiten buchen.

**Details**:

- Administratoren können spezifische Zeitblöcke oder Wochentage als Stoßzeiten markieren.

- Sie können Rabattraten für diese Zeiten festlegen, entweder in Prozent oder als feste Beträge.

- Kunden sehen die Rabatte deutlich auf der Buchungsseite und können den reduzierten Preis sehen.

**Vorteile**:

- Das Unternehmen kann die Auslastung in Zeiten mit niedriger Kundenfrequenz steigern.

- Kunden werden durch Rabatte dazu ermutigt, Termine außerhalb der Stoßzeiten zu buchen.

- Die Rabattfunktion erhöht die Flexibilität und die Möglichkeit zur Umsatzsteigerung.

# Masken und Controls

## Anmelde-Maske

* Benutzername-Feld
* Passwort-Feld
* Anmelde-Button
* Passwort vergessen-Link

## Administrations-Dashboard

* Übersicht über Mitarbeiter
* Verwaltung von Mitarbeitern (Hinzufügen, Bearbeiten, Löschen)
* Übersicht über Kunden und Buchungen
* Rabattverwaltung

## Kunden-Dashboard

* Terminbuchungskalender
* Terminauswahl
* Buchungsformular
* Terminverwaltung (Bearbeiten, Stornieren)
* Rabattinformationen

## Mitarbeiter-Dashboard

* Verfügbarkeitskalender
* Buchungsanfragen
* Annahme/Ablehnung von Buchungen
* Terminverwaltung
* Einnahmeübersicht

# Detailbeschreibungen für die Entwicklung:

## Anmelde-Maske:

- Benutzername-Feld: Ein einfaches Textfeld, in das der Benutzer seinen Benutzernamen eingibt.

- Passwort-Feld: Ein Passwortfeld, in das der Benutzer sein Passwort eingibt (Text wird maskiert).

- Anmelde-Button: Ein Button, um den Anmeldevorgang auszulösen.

- Passwort vergessen-Link: Ein Link, über den Benutzer ein Passwortwiederherstellungsverfahren starten können.

## Administrations-Dashboard:

- Übersicht über Mitarbeiter: Eine Tabelle oder Liste, die die Mitarbeiter und ihre grundlegenden Informationen anzeigt.

- Verwaltung von Mitarbeitern: Eine Funktion zur Hinzufügung, Bearbeitung und Löschung von Mitarbeitern, die Formulare für Mitarbeiterdaten verwendet.

- Übersicht über Kunden und Buchungen: Eine Übersicht über Kunden und deren gebuchte Termine.

- Rabattverwaltung: Eine Funktion zur Festlegung von Rabattzeiten und -beträgen.

## Kunden-Dashboard:

- Terminbuchungskalender: Ein Kalender, der die Verfügbarkeit der Mitarbeiter anzeigt.

- Terminauswahl: Ein Steuerelement, mit dem Kunden verfügbare Termine auswählen können.

- Buchungsformular: Ein Formular, in dem Kunden ihre Buchungsdetails eingeben können.

- Terminverwaltung: Eine Übersicht über gebuchte Termine, mit der Möglichkeit, Termine zu bearbeiten oder zu stornieren.

- Rabattinformationen: Die Anzeige von Rabatten für Termine in Stoßzeiten.

## Mitarbeiter-Dashboard:

- Verfügbarkeitskalender: Ein Kalender, der die Verfügbarkeit des Mitarbeiters anzeigt.

- Buchungsanfragen: Eine Liste der Buchungsanfragen, die der Mitarbeiter erhalten hat.

- Annahme/Ablehnung von Buchungen: Optionen, um Buchungsanfragen anzunehmen oder abzulehnen.

- Terminverwaltung: Eine Übersicht über gebuchte Termine und die Möglichkeit zur Bearbeitung.

- Einnahmeübersicht: Informationen über die Einnahmen des Mitarbeiters.

Die Detailbeschreibungen dienen als Grundlage für die Entwicklung der einzelnen Masken und Steuerelemente. Jedes Element muss entsprechend den Anforderungen und Benutzererwartungen entwickelt werden, um eine reibungslose Funktionsweise der Terminverwaltungswebanwendung sicherzustellen.

# Todos für den Datenbankentwickler im Zusammenhang mit der Erstellung der MongoDB-Datenbank für die Terminverwaltungswebanwendung:

## Todos für den Datenbankentwickler:

1. Datenmodell entwerfen: Entwerfen Sie ein Datenmodell, das die Anforderungen der Anwendung widerspiegelt. Berücksichtigen Sie die Entitäten wie Mitarbeiter, Kunden, Termine und Rabatte.

2. Schema für Mitarbeiter erstellen: Erstellen Sie ein Schema für Mitarbeiterdaten, das Informationen wie Name, Arbeitszeiten und Kontaktdaten enthält.

3. Schema für Kunden erstellen: Erstellen Sie ein Schema für Kundendaten, einschließlich persönlicher Informationen und Buchungsverlauf.

4. Schema für Termine erstellen: Entwerfen Sie ein Schema für Termindaten, das Informationen wie Zeitpunkt, Dauer, Mitarbeiterzuordnung und Kundenbuchung enthält.

5. Schema für Rabatte erstellen: Definieren Sie ein Schema für Rabatte, das Informationen über Zeitfenster und Rabattraten enthält.

6. Beziehungen zwischen den Schemas herstellen: Definieren Sie Beziehungen zwischen den Schemas, z. B. welche Termine von welchen Mitarbeitern gehalten werden und welche Rabatte für welche Zeiten gelten.

7. Indizes hinzufügen: Fügen Sie geeignete Indizes zu den Schemas hinzu, um die Abfrageleistung zu optimieren.

8. Testdaten generieren: Erstellen Sie Beispieldaten für Mitarbeiter, Kunden, Termine und Rabatte, um die Entwicklung und Tests zu erleichtern.

9. Datenbankverbindung einrichten: Konfigurieren Sie die Verbindung zur MongoDB-Datenbank in der Anwendung.

Nachdem die Todos abgeschlossen sind, kann man SQL-ähnliche Scripts verwenden, um die erforderlichen Tabellen und Relationen in MongoDB zu erstellen. MongoDB verwendet jedoch kein SQL, sondern BSON (Binary JSON) für die Datenbankabfragen. Hier ist ein Beispiel für MongoDB-Shell-Skripte zur Erstellung von Schemas und Beziehungen:

```javascript

// Schema für Mitarbeiter

db.createCollection("employees", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["name", "workingHours"],

properties: {

name: {

bsonType: "string",

description: "Name des Mitarbeiters"

},

workingHours: {

bsonType: "array",

description: "Arbeitszeiten des Mitarbeiters"

},

// Weitere Mitarbeiterattribute hier

}

}

}

});

// Schema für Kunden

db.createCollection("customers", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["name", "email"],

properties: {

name: {

bsonType: "string",

description: "Name des Kunden"

},

email: {

bsonType: "string",

description: "E-Mail-Adresse des Kunden"

},

// Weitere Kundenattribute hier

}

}

}

});

// Schema für Termine

db.createCollection("appointments", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["datetime", "employeeId", "customerId"],

properties: {

datetime: {

bsonType: "date",

description: "Datum und Uhrzeit des Termins"

},

employeeId: {

bsonType: "objectId",

description: "ID des Mitarbeiters"

},

customerId: {

bsonType: "objectId",

description: "ID des Kunden"

},

// Weitere Terminattribute hier

}

}

}

});

// Schema für Rabatte

db.createCollection("discounts", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["startDateTime", "endDateTime", "discountRate"],

properties: {

startDateTime: {

bsonType: "date",

description: "Startdatum und -zeit des Rabatts"

},

endDateTime: {

bsonType: "date",

description: "Enddatum und -zeit des Rabatts"

},

discountRate: {

bsonType: "double",

description: "Rabattrate (z.B. 0.2 für 20%)"

},

// Weitere Rabattattribute hier

}

}

}

});

// Beziehungen zwischen Schemas definieren

db.employees.createIndex({ "\_id": 1 });

db.customers.createIndex({ "\_id": 1 });

db.appointments.createIndex({ "employeeId": 1, "customerId": 1 });

db.discounts.createIndex({ "startDateTime": 1, "endDateTime": 1 });

```

Diese MongoDB-Shell-Skripte erstellen Sammlungen (Äquivalent zu Tabellen) für Mitarbeiter, Kunden, Termine und Rabatte. Sie definieren auch die erforderlichen Validierungsschemata und Indizes für eine effiziente Datenbanknutzung.

# Einige Todos für den Softwareentwickler im Zusammenhang mit der Entwicklung der Terminverwaltungswebanwendung in Java:

## Todos für den Softwareentwickler:

1. Projektstruktur festlegen: Legen Sie die grundlegende Projektstruktur und das Build-System (z.B. Gradle oder Maven) fest.

2. Abhängigkeiten konfigurieren: Fügen Sie die erforderlichen Bibliotheken und Abhängigkeiten zu Ihrem Projekt hinzu, z.B. Spring Boot, MongoDB Treiber, usw.

3. Datenbankanbindung einrichten: Konfigurieren Sie die Verbindung zur MongoDB-Datenbank, einschließlich Host, Port, Benutzername und Passwort.

4. Benutzer-Authentifizierung implementieren: Implementieren Sie die Benutzer-Authentifizierung für Administratoren, Kunden und Mitarbeiter, z.B. mit Spring Security.

5. Datenmodelle definieren: Erstellen Sie Java-Klassen, um die Datenmodelle für Mitarbeiter, Kunden, Termine und Rabatte darzustellen.

6. Repository-Klassen erstellen: Erstellen Sie Repository-Klassen, um auf die MongoDB-Datenbank zuzugreifen und CRUD-Operationen für die Datenmodelle durchzuführen.

7. Controller-Klassen entwickeln: Entwickeln Sie Controller-Klassen für die verschiedenen Benutzerrollen (Administrator, Kunde, Mitarbeiter), um die Anfragen zu verarbeiten.

8. Service-Klassen erstellen: Erstellen Sie Service-Klassen, um die Geschäftslogik zu implementieren, z.B. die Verwaltung von Terminen und Rabatten.

9. Validierungen hinzufügen: Fügen Sie Validierungen hinzu, um sicherzustellen, dass die eingegebenen Daten gültig sind, z.B. für Buchungen oder Mitarbeiterdaten.

10. Sicherheit implementieren: Implementieren Sie Sicherheitsmaßnahmen, um die Anwendung vor unberechtigtem Zugriff und Datenlecks zu schützen.

11. Benutzeroberfläche entwickeln: Erstellen Sie die Benutzeroberfläche (Front-End) für die Anwendung, wobei Sie Frameworks wie Spring MVC und Thymeleaf oder ein Front-End-Framework Ihrer Wahl verwenden können.

12. REST-APIs entwickeln: Wenn erforderlich, erstellen Sie RESTful APIs für die Kommunikation zwischen Front-End und Back-End.

13. Testfälle schreiben: Schreiben Sie Unit-Tests und Integrationstests, um sicherzustellen, dass die Anwendung korrekt funktioniert und stabile Ergebnisse liefert.

14. Dokumentation erstellen: Erstellen Sie eine Dokumentation für Entwickler, um den Code zu erklären und wie er verwendet werden soll.

15. Deployment vorbereiten: Bereiten Sie die Anwendung für das Deployment auf einer Produktionsumgebung vor, einschließlich Konfiguration von Datenbankverbindungen und Sicherheitsmaßnahmen.

## Java-Klassen und Funktionen:

Struktur für die Java-Klassen und Funktionen, die für die Terminverwaltungswebanwendung benötigt werden:

## 1. Datenmodelle:

- `**Employee.java**`: Klasse zur Darstellung von Mitarbeiterdaten.

- `**Customer.java**`: Klasse zur Darstellung von Kundendaten.

- `**Appointment.java**`: Klasse zur Darstellung von Terminen.

- `**Discount.java**`: Klasse zur Darstellung von Rabatten.

## 2. Repository-Klassen:

- `EmployeeRepository.java`: Repository-Klasse für die Datenbankzugriffe auf Mitarbeiterdaten.

- `CustomerRepository.java`: Repository-Klasse für die Datenbankzugriffe auf Kundendaten.

- `AppointmentRepository.java`: Repository-Klasse für die Datenbankzugriffe auf Termindaten.

- `DiscountRepository.java`: Repository-Klasse für die Datenbankzugriffe auf Rabattdaten.

## 3. Controller-Klassen:

- `AdminController.java`: Controller-Klasse für Administrator-Funktionen.

- `CustomerController.java`: Controller-Klasse für Kunden-Funktionen.

- `EmployeeController.java`: Controller-Klasse für Mitarbeiter-Funktionen.

## 4. Service-Klassen:

- `EmployeeService.java`: Service-Klasse für die Geschäftslogik im Zusammenhang mit Mitarbeitern.

- `CustomerService.java`: Service-Klasse für die Geschäftslogik im Zusammenhang mit Kunden.

- `AppointmentService.java`: Service-Klasse für die Geschäftslogik im Zusammenhang mit Terminen.

- `DiscountService.java`: Service-Klasse für die Geschäftslogik im Zusammenhang mit Rabatten.

## 5. Sicherheits-Klassen:

- `SecurityConfig.java`: Konfigurationsklasse für die Sicherheit der Anwendung.

- `CustomUserDetailsService.java`: Implementiert den benutzerdefinierten Benutzerdetaildienst für die Authentifizierung.

Diese Struktur dient als Ausgangspunkt und kann je nach den spezifischen Anforderungen der Anwendung erweitert und angepasst werden. Bitte beachten Sie, dass dies nur eine Beispielstruktur ist und die tatsächliche Implementierung von den Anforderungen und Präferenzen Ihres Projekts abhängt.

# Hier sind Epics (große, übergreifende Aufgaben) für die zuvor erstellten User Stories für die Terminverwaltungswebanwendung:

## Administratoren:

**1. Benutzer- und Mitarbeiterverwaltung:**

- Epic: Verwalten von Benutzern und Mitarbeitern

- Beschreibung: Dieses Epic konzentriert sich auf die Erstellung, Bearbeitung und Verwaltung von Benutzerkonten (Administratoren, Kunden und Mitarbeiter) sowie auf die Verwaltung der Mitarbeiterdaten.

**2. Rabattverwaltung:**

- Epic: Rabattverwaltung für Stoßzeiten

- Beschreibung: Dieses Epic konzentriert sich auf die Einrichtung und Verwaltung von Rabatten für Stoßzeiten sowie auf die Anzeige von Rabatten für Kunden während der Buchung.

## Kunden:

1**. Buchung und Verwaltung von Terminen:**

- Epic: Buchung und Verwaltung von Terminen

- Beschreibung: Dieses Epic ermöglicht Kunden die Buchung von Terminen, die Anzeige von Buchungen und die Verwaltung von Terminen, einschließlich Bearbeitung und Stornierung.

## Mitarbeiter:

1. Verfügbarkeitsverwaltung und Buchungsannahme:

- Epic: Verwaltung der Verfügbarkeit und Annahme von Buchungen

- Beschreibung: Dieses Epic konzentriert sich auf die Verwaltung der Verfügbarkeit des Mitarbeiters, die Annahme oder Ablehnung von Buchungen und die Verwaltung der eigenen Termine.

Diese Epics helfen dabei, die verschiedenen Aufgaben und Funktionen der Anwendung zu organisieren und ermöglichen eine bessere Planung und Priorisierung der Entwicklung. Sie können als übergreifende Aufgaben oder Projektabschnitte betrachtet werden, die in kleinere User Stories und Aufgaben aufgeteilt werden können, um die Umsetzung zu erleichtern.

# Entwicklung der Terminverwaltungswebanwendung

**1. Projektsetup und Planung:**

- Dies umfasst das Einrichten des Entwicklungsumfelds, die Festlegung von Projektzielen und -anforderungen sowie die Erstellung eines Entwicklungs- und Testplans.

**2. Datenbankdesign und -entwicklung:**

- Beginnen Sie mit der Erstellung der Datenbankstruktur, einschließlich der Definition von Schemas und Beziehungen zwischen den Datenmodellen. Implementieren Sie auch die Datenbankanbindung.

**3. Benutzer-Authentifizierung und Sicherheit:**

- Implementieren Sie die Benutzer-Authentifizierung und die Sicherheitsmaßnahmen für die Anwendung, um unberechtigten Zugriff zu verhindern.

**4. Datenmodelle und Repository-Klassen:**

- Erstellen Sie die Java-Klassen für Datenmodelle und Repository-Klassen für den Zugriff auf die MongoDB-Datenbank.

**5. Controller-Klassen:**

- Entwickeln Sie die Controller-Klassen für die verschiedenen Benutzerrollen (Administrator, Kunde, Mitarbeiter), um die Anfragen zu verarbeiten.

**6. Service-Klassen und Geschäftslogik:**

- Implementieren Sie die Service-Klassen, um die Geschäftslogik zu entwickeln, wie z.B. Buchungen, Verfügbarkeit und Rabatte.

7. **Benutzeroberfläche (Front-End):**

- Entwickeln Sie die Benutzeroberfläche (Front-End) für die Anwendung, einschließlich der Benutzeroberflächenkomponenten und -seiten.

8. **REST-APIs (falls erforderlich):**

- Implementieren Sie RESTful APIs, falls die Anwendung eine API für die Kommunikation zwischen Front-End und Back-End benötigt.

**9. Validierungen und Tests:**

- Fügen Sie Validierungen für Formulare und Eingaben hinzu und schreiben Sie Unit-Tests und Integrationstests, um die Funktionalität sicherzustellen.

**10. Sicherheitstests und Penetrationstests:**

- Überprüfen Sie die Anwendung auf Sicherheitslücken und führen Sie Penetrationstests durch, um Schwachstellen zu identifizieren und zu beheben.

**11. Dokumentation und Benutzerhandbuch:**

- Erstellen Sie Entwicklerdokumentation und Benutzerhandbuch, um die Verwendung der Anwendung zu erleichtern.

12. **Deployment und Produktionsvorbereitung:**

- Bereiten Sie die Anwendung für das Deployment in einer Produktionsumgebung vor, einschließlich der Konfiguration von Datenbankverbindungen und Sicherheitsmaßnahmen.

13. **Schulung und Einführung:**

- Führen Sie Schulungen für die Benutzer und Administratoren durch und leiten Sie sie in der Anwendungsnutzung ein.

14. **Überprüfung und Qualitätssicherung:**

- Führen Sie abschließende Überprüfungen und Qualitätssicherungsprozesse durch, um sicherzustellen, dass die Anwendung ordnungsgemäß funktioniert.

15. **Inbetriebnahme und Start:**

- Nehmen Sie die Anwendung in Betrieb und starten Sie den produktiven Einsatz.

Diese Reihenfolge dient als grobe Richtlinie, und es kann notwendig sein, Aufgaben zu parallelisieren oder in Abhängigkeit von den spezifischen Anforderungen Ihres Projekts anzupassen. Es ist wichtig, während des gesamten Projekts auf die Prioritäten zu achten und regelmäßig Updates und Überprüfungen durchzuführen, um sicherzustellen, dass das Projekt im Zeitplan bleibt und die Anforderungen erfüllt werden.

Ein Meilensteinplan für 32 Wochen kann helfen, die wichtigsten Etappen und Ziele für Ihr Projekt im Laufe der Zeit zu organisieren. Hier ist ein Beispiel für einen Meilensteinplan:

# Terminplan

**Woche 1-2: Projektstart und Planung**

- Projektinitialisierung

- Anforderungsanalyse und Spezifikation

- Teamzusammenstellung und Ressourcenplanung

- Erstellung eines Projektplans und Zeitplans

- Beginn der Datenbankentwicklung

**Woche 3-4: Benutzer-Authentifizierung und Sicherheit**

- Implementierung der Benutzer-Authentifizierung

- Einrichtung von Sicherheitsmaßnahmen

- Entwicklung von Validierungen und Authentifizierungstests

**Woche 5-6: Datenbankentwicklung**

- Fortsetzung der Datenbankentwicklung

- Erstellung der MongoDB-Schemas

- Implementierung der Repository-Klassen

- Integrationstests für die Datenbank

**Woche 7-8: Controller und Service-Klassen**

- Entwicklung der Controller-Klassen

- Implementierung der Service-Klassen und Geschäftslogik

- Beginn der Front-End-Entwicklung

**Woche 9-10: Benutzeroberfläche (Front-End)**

- Fortsetzung der Front-End-Entwicklung

- Gestaltung von Benutzeroberflächenkomponenten und -seiten

- Integration mit Back-End-Services

**Woche 11-12: REST-APIs (falls erforderlich)**

- Implementierung von RESTful APIs

- Schnittstellenentwicklung und Dokumentation

- Validierung und Tests der APIs

**Woche 13-16: Buchungsfunktionen und Kundenverwaltung**

- Implementierung der Buchungsfunktionen für Kunden

- Verwaltung von Kundendaten

- Buchungstests und Validierung

**Woche 17-20: Mitarbeiterverwaltung und Verfügbarkeit**

- Entwicklung von Mitarbeiterverwaltungsfunktionen

- Verfügbarkeitsverwaltung für Mitarbeiter

- Test und Validierung der Mitarbeiterfunktionen

**Woche 21-24: Rabattverwaltung und Sicherheitstests**

- Implementierung der Rabattverwaltung für Stoßzeiten

- Sicherheitstests und Penetrationstests durchführen

- Behebung von Sicherheitslücken und Schwachstellen

**Woche 25-28: Dokumentation und Benutzerhandbuch**

- Erstellung von Entwicklerdokumentation

- Erstellung eines Benutzerhandbuchs

- Schulungsvorbereitung für Benutzer und Administratoren

**Woche 29-30: Produktionsvorbereitung und Deployment**

- Vorbereitung der Anwendung für das Produktionsdeployment

- Konfiguration von Datenbankverbindungen und Sicherheitsmaßnahmen

**Woche 31: Inbetriebnahme und Schulung**

- Inbetriebnahme der Anwendung

- Schulungen für Benutzer und Administratoren

Woche 32: Projektabschluss und Qualitätssicherung

- Letzte Überprüfung und Qualitätssicherung

- Abschlussbericht und Projektdokumentation

- Projektabschluss und Übergabe an die Stakeholder

Bitte beachten Sie, dass dies nur ein Beispiel für einen Meilensteinplan ist, und Sie sollten die Zeiten und Aufgaben entsprechend den spezifischen Anforderungen und Ressourcen Ihres Projekts anpassen. Es ist auch wichtig, regelmäßig den Fortschritt zu überprüfen und den Zeitplan bei Bedarf anzupassen.