

FH Münster
Fachbereich Elektrotechnik und Informatik

Pirate Bay Tours Booking Client

Dokumentation

Gruppe 4

Lars Alexander Naujoks - ln464351@fh-muenster.de - Matr.-Nr.: 1121632

Alexander Preckel - ap369794@fh-muenster.de - Matr.-Nr.: 1129922

Jan Volker Röhrig - jr911076@fh-muenster.de - Matr.-Nr.: 1140248

Antonio Sarcevic - as019568@fh-muenster.de - Matr.-Nr.: 1130018

Rebecca Zyla - rz801925@fh-muenster.de - Matr.-Nr.: 1141331

Dozent

Prof. Dr.-Ing. Thomas Christian Weik

Inhaltsverzeichnis

1	Einleitung	1
1.1	Aufgabenbeschreibung	1
2	Umsetzung	2
2.1	Lösungsansatz	2
2.2	Datenstruktur	4
2.3	Verwendete Technologien	8
3	Fazit	13
A	Benutzerhandbuch	14
A.1	Starten der Anwendung und von CouchDB	14
A.2	Einloggen	15
A.3	Tour erstellen	16
A.4	Sync-Button	17
A.5	Reservierungen hinzufügen	18
A.6	Löschen von Touren	20
A.7	Drucken der Liste der Reservierungen	20
A.8	Extras	22

1 Einleitung

Die folgende Projektdokumentation handelt von dem Projekt „Pirate Bay Tours“, welches im Zuge des Moduls „Verteilte Informationssysteme“ bearbeitet wurde.

„Pirate Bay Tours“ beschreibt hierbei ein Unternehmen, welches Kreuzfahrten in der Karibik anbietet. Für das Unternehmen arbeiten mehrere Agenten, welche tagsüber Reservierungen für die unterschiedlichen Touren entgegennehmen.

Ziel war es, das Projekt in Gruppen von vier bis fünf Personen umzusetzen. Hierfür haben sich in unserer Gruppe Lars Alexander Naujoks, Alexander Preckel, Jan Volker Röhring, Antonio Sarcevic und Rebecca Zyla zusammengefunden.

1.1 Aufgabenbeschreibung

Die Aufgabe bestand darin, eine Anwendung zu entwickeln, welche das Anlegen und Verwalten von Touren ermöglicht. Folgende Anforderungen wurden an die Anwendung gestellt:

- Anwendung dient den Agenten, um Touren anzulegen und neue Reservierungen entgegenzunehmen.
- Mehrere Kreuzfahrten mit unterschiedlichen Booten und daher variierenden Kapazitäten sind möglich.
- Agenten haben nicht immer eine Internetverbindung und arbeiten daher oft offline.
- Zu bestimmten Zeiten synchronisieren sich alle Agenten mit dem Server.

Die Gruppe hat für die Erfüllung dieser Aufgabe den „Pirate Bay Tours Booking Client“ erstellt.

2 Umsetzung

In diesem Abschnitt wird die Umsetzung des Pirate Bay Tours Booking Client besprochen. Hierfür wird zunächst der Algorithmus für die Durchführung einer Buchung unter Berücksichtigung der Vorgaben erläutert. Anschließend wird die Datenstruktur, die sich aus dem Lösungsansatz ergibt, erklärt. Zuletzt werden die zur Umsetzung verwendeten Technologien eingeführt und ihre Verwendung innerhalb der Anwendung erläutert.

2.1 Lösungsansatz

Die Grundidee des Buchungsalgorithmus ist, jedem Sales Agent ein bestimmtes Kontingent an offline Buchungen zuzusprechen. Das bedeutet, dass Sales Agents bis zu einer bestimmten Anzahl an Gästen Buchungen ohne Synchronisation durchführen und direkt dem Kunden bestätigen können. Hat ein Sales Agent sein Kontingent aufgebraucht, wird die Buchung nicht direkt als fehlgeschlagen geachtet, sondern wird zurückgehalten. Zurückgehaltene Buchungen werden dann jede Nacht von einem serverseitigem Skript bearbeitet. Der Sales Agent bekommt dann am nächsten Morgen eine Meldung, dass seine vorher zurückgehaltene Buchung bearbeitet wurde, damit er den Kunden bestätigen oder absagen kann. Der Algorithmus zur Synchronisation der Buchungen und Touren besteht aus drei Teilen:

1. Sync: Wenn ein Sales Agent eine aktive Internetverbindung hat und den „sync now“-Button in der Webanwendung betätigt. Dies soll von jedem Sales Agent zu Arbeitsbeginn und -ende getan werden.
2. Buchung: Wenn der Sales Agent eine neue Buchung zu einer Tour in der Webanwendung hinzugefügt.
3. Bearbeiten zurückgehaltener Buchungen: Zurückgehaltene Buchungen werden Nachts von einem Skript bearbeitet. Dieses Skript läuft serverseitig ab.

Teil 1: Sync

Wenn ein Sales Agent den „sync now“-Button in der Webanwendung betätigt werden zunächst die lokalen PouchDB Datenbanken der Webanwendung mit der zentralen, ausgelagerten CouchDB Datenbank synchronisiert. Dies gibt der Webanwendung zugriff auf alle vorher synchronisierten Touren und Buchungen und spiegelt mögliche Änderungen, die in der Webanwendung vorgenommen wurden, auf der ausgelagerten CouchDB Datenbank.

Nachdem die Datenbanken synchron sind, wird in der Webanwendung das Kontingent für den Sales Agent berechnet. Das Kontingent ergibt sich aus der Differenz zwischen der Anzahl an Plätze, die für eine Tour vorgesehen sind, und den bereits vorhandenen Buchungen. Diese Zahl

wird dann durch die Anzahl der Sales Agenten geteilt, welche als Konstante im Quellcode vorliegt. Dies geschieht für jede Tour. Außerdem wird pro Tour die Hilfsvariable, die zählt wie viele Plätze bereits offline vergeben wurden auf 0 gesetzt. Der Algorithmus zur Generierung des Kontingents wird noch einmal als Pseudocode dargestellt:

```
for each tour in tours
    allowance = (tour.seats - tour.reservedSeats) / numberOfAgents
    bookedSinceLastSync = 0
```

Im letzten Schritt werden die vorher zurückgehaltenen Buchungen mit nun bearbeiteten Buchungen verglichen. Findet der Client vorher zurückgehaltene Buchungen, die nun bearbeitet wurden, werden diese als Notiz oben in der Webanwendung angezeigt. Dieser Teil des Algorithmus wird zu Arbeitsbeginn und -ende vom Sales Agent innerhalb der Webanwendung ausgeführt.

Teil 2: Buchung

Fügt ein Sales Agent, innerhalb der Webanwendung, eine neue Buchung hinzu, wird dieser Teil des Algorithmus durchgeführt. Dieser Teil bestimmt, ob eine Buchung direkt als offline Buchung durchgeführt werden kann oder ob die Buchung zurückgehalten werden muss.

Hierfür wird das im Teil 1 generierte Kontingent mit der Zahl der erlaubten Überbuchung multipliziert, welche ähnlich wie die Anzahl der Sales Agents im Quellcode festgelegt ist. Falls diese Zahl größer gleich als die Anzahl der bereits offline vergebenen Plätze summiert mit der Gruppengröße dieser Buchung ist, kann die Buchung in die Tour geschrieben werden. Außerdem wird noch die Anzahl der bereits offline vergebenen Plätze um die Anzahl der Gruppengröße dieser Buchung erhöht. Falls nicht, wird die Buchung zurückgehalten, damit sie von einer zentralen Instanz bearbeitet werden kann. Folgendes Nassi-Shneiderman-Diagramm erläutert den Ablauf des Algorithmus zur Erstellung einer Buchung:

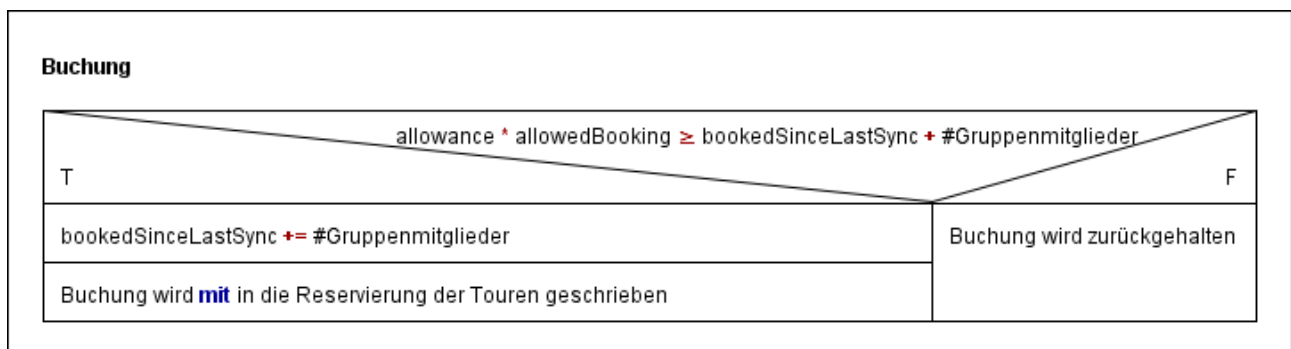


Abbildung 1: Nassi-Shneiderman-Diagramm zur Durchführung einer Buchung

Teil 3: Bearbeitung zurückgehaltener Buchungen

Der dritte Teil des Algorithmus bearbeitet alle Buchungen, die nicht offline gebucht werden konnten und zurückgehalten wurden. Dieser Teil wird jede Nacht von einem serverseitigem Skript (im folgenden `pbScript`) durchgeführt.

Hier wird für jede zurückgehaltene Buchung berechnet, ob die Anzahl der Plätze einer Tour multipliziert mit der erlaubten Überbuchung größer gleich der Anzahl der bereits reservierten Plätze plus die Gruppengröße ist. Ist dies der Fall, kann die Buchung zur Tour hinzugefügt werden und gilt als erfolgreich. Ist dies nicht der Fall würde die Buchung die verfügbaren Plätze zu sehr überschreiten, kann nicht hinzugefügt werden und gilt als abgelehnt.

In beiden Fällen wird ein Eintrag in die ausgelagerte CouchDB Datenbank geschrieben, die von der Webanwendung genutzt wird, um die Sales Agents über vorher zurückgehaltenen Buchungen zu informieren. Folgendes Nassi-Shneiderman-Diagramm erläutert den Ablauf des Algorithmus zum Bearbeiten einer zurückgehaltenen Buchung:



Abbildung 2: Nassi-Shneiderman-Diagramm zur Bearbeitung einer zurückgehaltenen Buchung

2.2 Datenstruktur

Aus dem in Kapitel 2.1 besprochenem Buchungsalgorithmus wurde eine Datenbankstruktur entwickelt, die aus vier verschiedenen Teilen besteht:

1. `tours`: Enthält Touren und erfolgreiche Buchungen
2. `pendingBookings`: Enthält zurückgehaltenen Buchungen
3. `bookingsNeedContact`: Enthält vorher zurückgehaltene Buchungen mit Status
4. `tourAllowance`: Enthält Kontingente für jede Tour

Da bereits vorher entschieden war, dass CouchDB für die einfache Synchronisation von Webanwendung und Datenbank genutzt werden soll, war eine Objektstruktur erforderlich.

tours

Die Datenbank `tours` enthält alle Touren und dazugehörige erfolgreich durchgeführte Buchungen. Jede Tour besitzt neben einem `_id`-Feld noch die Felder `name`, welcher den Namen der Tour enthält, `description`, welche eine Beschreibung der Tour enthält, `maxPassenger`, welche die Anzahl der Plätze enthält und `reservations`, ein Array welches mit erfolgreich durchgeführten Buchungen gefüllt wird.

Einträge des `reservations` Array bestehen aus Objekten mit den Feldern `firstName`, `lastName`, welche zum Gruppenleiter gehören, und `groupSize`, welches die Anzahl der Gruppenmitglieder enthält. Ein Eintrag der Datenbank `tours` sieht beispielsweise wie folgt aus:

```
{
  "_id": "2020-12-02T15:38:22.633Z",
  "name": "tour name",
  "description": "tour description",
  "maxPassenger": 100,
  "reservations": [
    {
      "firstName": "Max",
      "lastName": "Muster",
      "groupSize": 2
    },
  ]
}
```

Diese Einträge können durch Erstellen einer Tour innerhalb der Webanwendung erzeugt werden. Diese Einträge werden beim Betätigen des „sync now“-Buttons von der lokalen Webanwendung auf die ausgelagerte CouchDB Datenbank gespiegelt.

Die Webanwendung kann außerdem noch erfolgreich durchgeführte offline Buchungen aus dem zweiten Teil des Buchungsalgorithmus in das `reservations`-Array schreiben. `pbScript` kann ebenfalls erfolgreich durchgeführte Buchungen aus dem dritten Teil des Buchungsalgorithmus in das `reservations`-Array schreiben.

pendingBookings

Die Datenbank `pendingBookings` enthält alle von der Webanwendung zurückgehaltenen Buchungen. Neben einem `_id`-Feld enthält jede zurückgehaltene Buchung noch ein `tourId`-Feld, welches der `_id` der Tour entspricht, die zu dieser Buchung gehört. Zusätzlich werden Informationen zum Gruppenleiter in den Feldern `firstName`, `lastName` und `phone` gespeichert, die be-

nötigt werden, damit der Gruppenleiter nach Bearbeiten der Buchung informiert werden kann. Die Anzahl der Gruppenmitglieder wird in dem Feld `groupSize` gespeichert. Ein Beispieleintrag der `pendingBookings` Datenbank sieht wie folgt aus:

```
{
  "_id": "2020-12-08T13:32:10.880Z",
  "tourId": "2020-12-02T15:38:22.633Z",
  "firstName": "Max",
  "lastName": "Muster",
  "phone": "+555420",
  "groupSize": 100
}
```

Einträge der Datenbank `pendingBookings` werden von der Webanwendung erstellt, wenn eine Buchung nicht offline durchgeführt werden kann. Diese Einträge werden beim Betätigen des „sync now“-Buttons von der lokalen Webanwendung auf die ausgelagerte CouchDB Datenbank gespiegelt. Das `pbScript` verwendet diese Einträge dann, um zurückgehaltene Buchungen zu bearbeiten.

bookingsNeedContact

Kunden, deren Buchungen zurückgehalten wurden, müssen nach Bearbeitung der Buchung vom Sales Agent über den Status der Buchung informiert werden. Hierzu wird ein Eintrag in der Datenbank `bookingsNeedContact` angelegt. Diese Einträge enthalten, neben dem `_id`-Feld, die Kontaktinformationen des Gruppenleiters in den Feldern `firstName`, `lastName` und `phone`. Zusätzlich gibt es noch ein `status`-Feld, in dem der Status der Buchung festgehalten wird, entweder `success` oder `declined`. Ein Eintrag sieht beispielsweise folgendermaßen aus:

```
{
  "_id": "2020-11-09T13:15:22.821Z",
  "firstName": "Sophie",
  "lastName": "Xeon",
  "phone": "+555420",
  "status": "declined"
}
```

Diese Einträge werden von `pbScript` nach dem Bearbeiten der zurückgehaltenen Buchung in Teil 3 des Buchungsalgorithmus erstellt. Diese Einträge werden nach einem Sync von der Web-

anwendung als Liste angezeigt, um dem Sales Agent zu signalisieren, dass er diesen Kunden kontaktieren muss.

tourAllowance

Die Datenbank `tourAllowance` wird genutzt um für jede Tour die Hilfsvariablen `allowance`, also das Kontingent, und `bookedSinceLastSync` zu sichern. Diese werden benötigt, um die Durchführung von offline Buchungen zu ermöglichen. Das `_id`-Feld entspricht hier dem `_id`-Feld der zugehörigen Buchung. Ein Objekt in dieser Datenbank kann beispielsweise wie folgt aussehen:

```
{
  "_id": "2020-12-02T15:38:22.633Z",
  "allowance": 5.8,
  "bookedSinceLastSync": 0
}
```

Einträge dieser Datenbank werden beim ersten Teil des Buchungsalgorithmus, dem Sync, für jede Tour neu generiert, jedoch nicht auf die ausgelagerte CouchDB synchronisiert, da jeder Sales Agent sein eigenes Kontingent besitzt.

Datenfluss

Folgendes Sequenzdiagramm soll beispielhaft den Datenfluss einer zurückgehaltenen Buchung, die erfolgreich bearbeitet wird, erläutern:

Der Sales Agent erstellt eine neue Buchung über das Interface der Webanwendung. Die Webanwendung überprüft, ob das Kontingent ausreicht, um diese Buchung durchzuführen. Das Kontingent reicht nicht aus, weshalb die Buchung in die `pendingBookings` Datenbank geschrieben wird. Der Sales Agent betätigt abends zum Arbeitsende den „sync now“-Button, wodurch alle lokalen Datenbanken mit der ausgelagerten CouchDB synchronisiert werden.

Das serverseitige Script wird nachts angestoßen und ruft die Einträge der `pendingBookings` Datenbank auf. Für jeden Eintrag wird berechnet, ob die Gruppe noch in die Tour geschrieben werden kann. Dies ist der Fall, weshalb die Buchung in die Tour geschrieben wird und ein Eintrag in der `bookingsNeedContact` erstellt wird. Die Änderungen werden mit der CouchDB synchronisiert.

Am nächsten Arbeitstag betätigt der Sales Agent zum Arbeitsbeginn den „sync now“-Button, wodurch erneut alle lokalen Datenbanken mit der ausgelagerten CouchDB synchronisiert werden. Die Webanwendung verarbeitet die Einträge aus `bookingsNeedContact` und zeigt dem

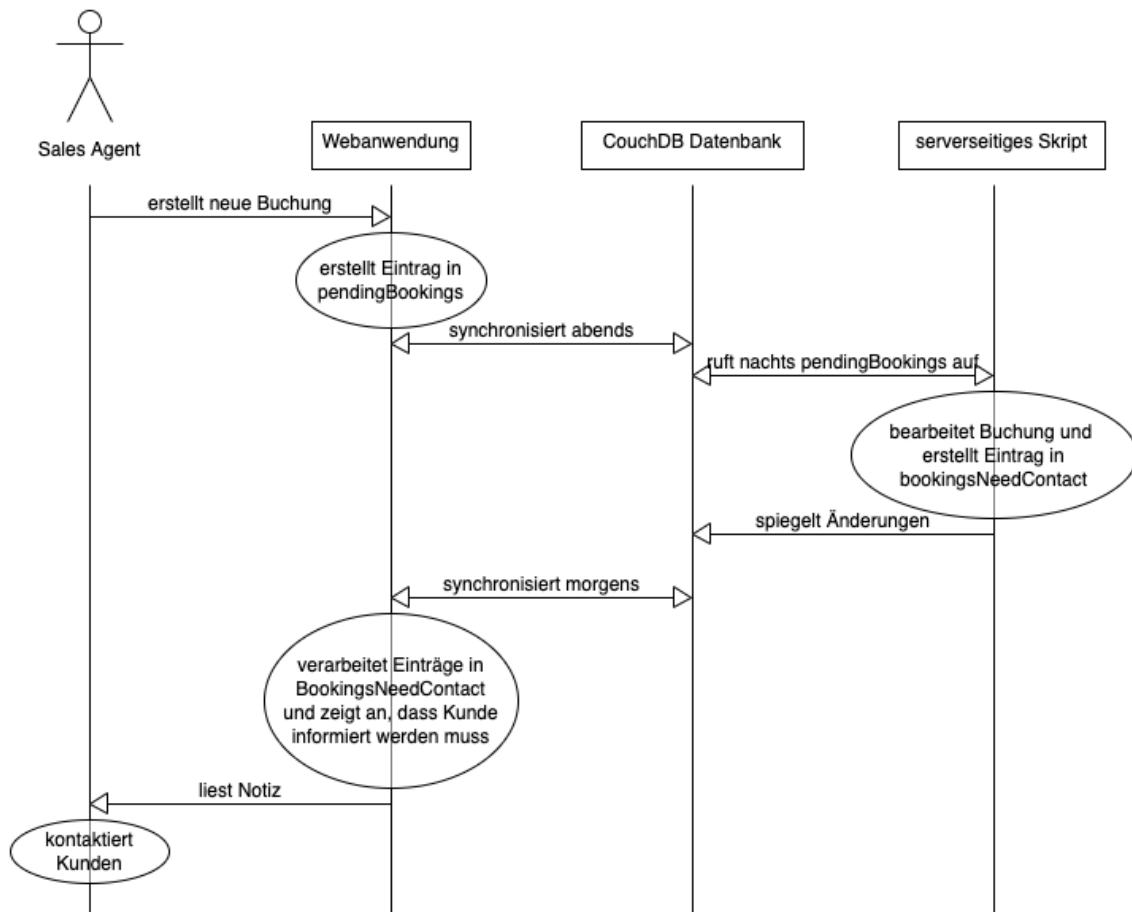


Abbildung 3: Sequenzdiagramm zur Durchführung einer erst zurückgehaltenen, dann erfolgreichen Buchung

Sales Agent an, dass eine Buchung, die vorher zurückgehalten wurde, nun durchgeführt wurde. Der Sales Agent kontaktiert den Kunden und informiert ihn über den Status der Buchung.

2.3 Verwendete Technologien

Innerhalb dieses Kapitels werden die für die Umsetzung verwendeten Technologien eingeführt und ihre Verwendung für die Anwendung erläutert.

Docker

Docker¹ ist eine Software zur Isolierung von Anwendungen vom Hostsystem durch Containervirtualisierung. Dies vereinfacht die Entwicklung und Auslieferung von Anwendungen, da alle benötigten Software Abhängigkeiten innerhalb von Containern installiert werden können. Ein Container ist eine abgekapselte Umgebung, die durch ein Container Image initialisiert wird. In diesem Image kann z. B. ein Softwarepaket installiert werden. Wird das Image als Container gestartet, kann das Softwarepaket in dem Container genutzt werden, auch wenn das Hostsystem,

¹<https://www.docker.com/>

auf dem der Container läuft, dieses Softwarepaket nicht besitzt oder inkompatibel damit ist. Vorgefertigte Images können vom Docker Hub geladen werden, um z. B. eine isolierte Node.js Entwicklungsumgebung aufzusetzen. Mehrere Container können mit einer `docker-compose.yml`-Datei als eine Applikation zusammengefasst werden. In dieser Datei wird unter anderem definiert, welche Container Images genutzt werden, welche Ports von außerhalb des Containers verfügbar sein sollen und welche Container Abhängigkeiten bestehen. Außerdem können sogenannte „Volumes“ genutzt werden, um bestimmte Ordner des Hostsystems in den Docker Container einzuhängen und so Daten auszutauschen.

Pirate Bay Tours Booking Client nutzt eine `docker-compose.yml`-Datei, um drei verschiedene Container zu einer Applikation zusammenzufassen. Im `couch` Container befindet sich der CouchDB Server, der genutzt wird, um die Daten zentralisiert zu sichern. Der `app` Container liefert die Webanwendung aus, während der `pbscript` Container genutzt wird, um das serverseitige Skript zum Bearbeiten der zurückgehaltenen Buchungen auszuführen.

Der `app` Container benötigt den freigeschalteten Port `8000`, damit ein Nutzer die Webanwendung aufrufen kann. Das zugehörige Docker Image kopiert den Quellcode aus dem `/app/` Ordner, installiert Abhängigkeiten und baut die Applikation. Anschließend wird ein Python HTTP Server gestartet, um die fertig gebaute Webanwendung auszuliefern.

Der Container für das `pbScript` welches die zurückgehaltenen Buchungen bearbeitet, benötigt ebenfalls für die Netzwerkkommunikation einen Port. Hier wurde der Port `8080` freigeschaltet. Innerhalb des Docker Images werden lediglich die benötigten Abhängigkeiten installiert.

CouchDB

CouchDB² ist eine quelloffene und Dokument-basierte Datenbank mit integriertem HTTP Server für einfachen Datenaustausch im JSON Format. Der große Vorteil von CouchDB gegenüber anderen Dokument-basierten Datenbanken ist, dass CouchDB ein integriertes Protokoll zur Synchronisation anbietet. Das bedeutet, dass einem CouchDB durch das „CouchDB Replication Protocol“ die Arbeit der Replikation von Daten auf verschiedenen Datenbanken abnimmt. Dies eignet sich besonders gut für die Implementierung von Offline-First-Anwendungen wie dem Pirate Bay Tours Booking Client. Wie genau das CouchDB Protokoll zur Replikation funktioniert, kann unter den CouchDB Docs nachgelesen werden³.

PouchDB⁴ ist eine quelloffene JavaScript Datenbank, die im Browser oder in einem NodeJS Server genutzt werden kann. PouchDB ist mit CouchDB kompatibel und nutzt ebenfalls das „CouchDB Replication Protocol“ zum einfachen replizieren und persistieren von Daten im Browser oder NodeJS Umgebungen.

²<https://couchdb.apache.org/>

³<https://docs.couchdb.org/en/stable/replication/protocol.html>

⁴<https://pouchdb.com/>

Für den Pirate Bay Tours Booking Client wird CouchDB als zentralisierter, ausgelagerter Datenbankserver genutzt, um die verschiedenen Sales Agents miteinander zu verknüpfen. Die Webanwendung und `pbScript` nutzen PouchDB, um mit dem CouchDB Server zu kommunizieren, die Daten zu synchronisieren und dann im Browser zu persistieren, damit eine offline Nutzung möglich ist.

CouchDB wird durch die `docker-compose.yml`-Datei und ein dazugehöriges Docker Image konfiguriert. Für den couch Container muss der Port 5984 nach außen freigegeben werden und zusätzlich, zur Persistierung der Daten, ein volume mapping angelegt werden. Das Docker Image für den couch Container basiert auf dem `couchdb:2.1.1` Image welches unter Docker Hub zu finden ist. Zusätzlich werden zwei Konfigurationen vorgenommen: CORS-Regeln werden ausgeschaltet, damit die Webanwendung auf die CouchDB Datenbank zugreifen kann, und es wird ein Admin-Nutzer angelegt.

NodeJS

NodeJS⁵ ist eine quelloffene JavaScript Laufzeitumgebung, die auf der V8 JavaScript Engine beruht. Es erlaubt das Ausführen von JavaScript außerhalb von einem Browser und ermöglicht dadurch, dass JavaScript für serverseitige Anwendungen genutzt werden kann. NodeJS wird für die Durchführung von `pbScript` genutzt, also dem dritten Teil des Buchungsalgorithmus. Hierzu werden lokale PouchDB Instanzen gestartet, die sich mit der entfernten CouchDB Instanz verbinden und die Daten einlesen. Das `pbScript` nutzt dann die Einträge in der `pendingBookings` Datenbank um zurückgehaltene Buchungen zu bearbeiten. Jeder Eintrag enthält eine `tourId`, welche genutzt wird, um die zugehörige Tour auszulesen. Dann wird zusammengezählt, wie viele Plätze bereits belegt sind, und falls noch genügend Plätze für die zurückgehaltenen Plätze verfügbar sind, ohne dass eine zu hohe Überbuchung stattfindet, kann die Buchung mit in die Tour geschrieben werden. Nachdem die Buchung bearbeitet wurde, erstellt `pbScript` einen Eintrag in `bookingsNeedContact`, damit die Webanwendung den Sales Agent über den neuen Status der Buchung informieren kann.

Svelte

Für das Frontend der Webanwendung kommt Svelte⁶ zum Einsatz. Svelte unterscheidet sich von anderen SPA Frameworks wie Angular, Vue und React dadurch, dass es ein Compiler ist und dadurch möglichst effizientes JavaScript produzieren kann. Bei Angular, Vue und React muss im Browser das gesamte Framework geladen werden, während Svelte durch den Bau-Schritt dies nicht benötigt. Die Vorteile sind also kleinere Bundle Sizes und bessere Performance, da

⁵<https://nodejs.org/>

⁶<https://svelte.dev/>

kein abgleichen mit einem Virtual DOM benötigt wird. Außerdem erlaubt der Svelte Compiler eine eigene Syntax, welche den Code simplifiziert. Svelte nutzt ähnlich wie die oben genannten Frameworks eine Komponentenstruktur, um eine Anwendung in kleinere Teile aufzuspalten. Im Folgenden wird die Komponentenstruktur vom Pirate Bay Tours Booking Client kurz dargestellt:

- App: Root Komponente, enthält das grundlegende Layout und versteckt die Tours Komponente, falls der Nutzer nicht eingeloggt ist
 - ThemeSwitcher: Erlaubt das Wechseln des verwendeten Farbthemas vom Pirate Bay Tours Booking Client
 - Login: Verarbeitet das einloggen und ausloggen von Nutzern mit dem PouchDB Plugin `pouchdb-auth`⁷
 - Tours: Führt die Synchronisation der Daten von CouchDB mit PouchDB durch, generiert die Kontingente und erstellt Notizen für Sales Agent basierend auf `bookingsNeedContact`-Einträgen (Teil 1 des Buchungsalgorithmus), handelt außerdem das Erstellen von Touren und Reservierungen und dem Löschen von Touren und den `bookingsNeedContact`-Notizen
 - ✦ Tour: Enthält die Struktur eines einzelnen Tour-Eintrags, stellt Tour Informationen dar und ermöglicht durch das npm-Paket `marked`⁸ das Anzeigen der Tourbeschreibung in Markdown und durch das npm-Paket `print-js`⁹ das Ausdrucken von Reservierungslisten.
 - ✦ CreateTour: Enthält das Formular zum Erstellen von neuen Touren und kümmert sich um Validation der Eingabe, ermöglicht ebenfalls durch `marked` das Anzeigen der Tourbeschreibung in Markdown
 - ✦ CreateReservation: Enthält das Formular zum Erstellen von neuen Reservierungen und kümmert sich um Validation der Eingabe
- Weitere Komponenten:
 - Modal: Wird von verschiedenen Komponenten genutzt, um ein dem Hauptfenster untergeordnetes Fenster zu öffnen
 - Error: Wird von Formularen genutzt, um den Nutzer auf fehlerhafte Eingaben hinzuweisen
 - Confirm: Wird genutzt, damit der Nutzer bestimmte Aktionen, wie das Löschen einer Tour, bestätigen muss

⁷<https://www.npmjs.com/package/pouchdb-auth>

⁸<https://marked.js.org/>

⁹<https://printjs.crabbly.com/>

Svelte wird also genutzt um den Sales Agents ein Frontend bereitzustellen. In Kombination mit Svelte wird PouchDB und pouchdb-auth genutzt, um die Datenkommunikation mit dem CouchDB Server zu ermöglichen. Weitere JavaScript Bibliotheken wie marked und printJS wurden genutzt, um extra Features bereitzustellen.

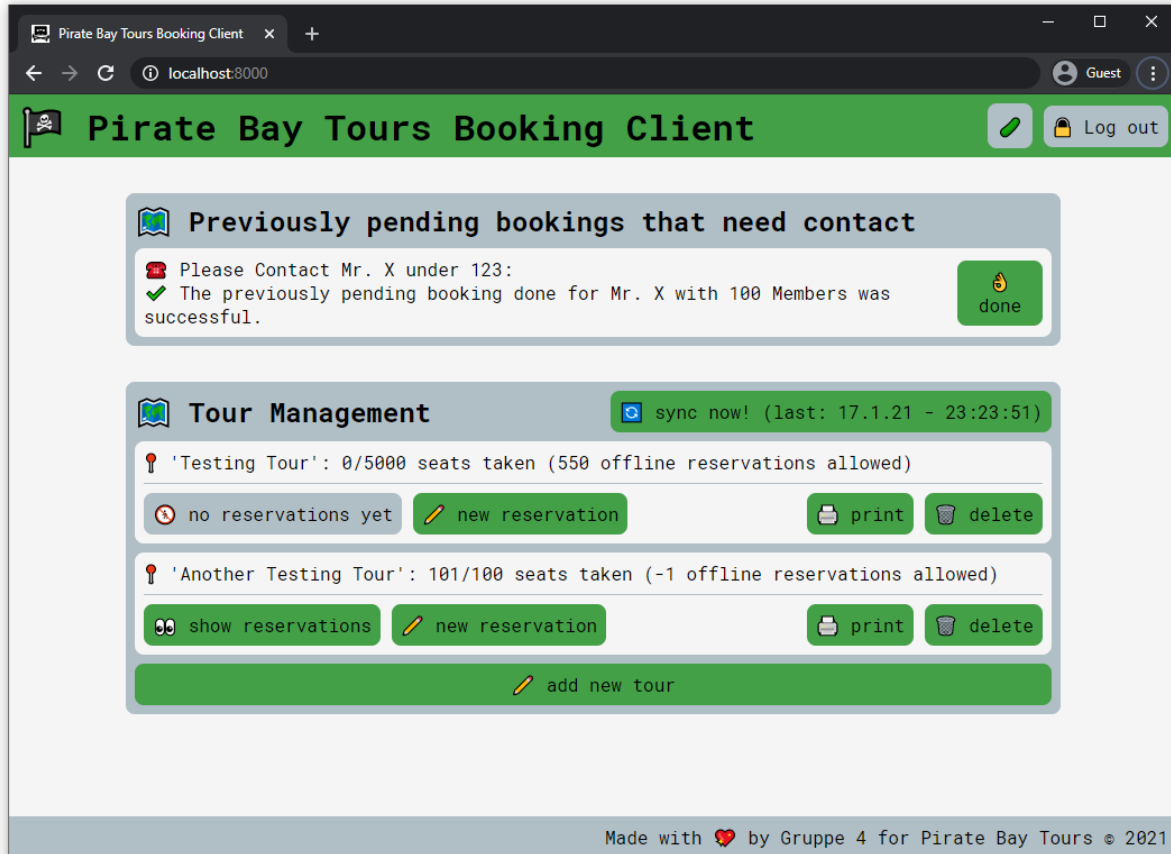


Abbildung 4: Webanwendung: Pirate Bay Tours Booking Client

3 Fazit

CouchDB hat sich für den Use Case einer simplen Offline-First-Anwendung bewährt. Durch Docker konnte die Konfiguration simplifiziert werden. Die Kommunikation mit den Clients durch PouchDB war kinderleicht zu konfigurieren. PouchDB speichert die Daten automatisch im Browser ab, sodass diese auch offline verfügbar sind. Durch pouchdb-auth konnte außerdem noch eine Nutzerauthentifizierung implementiert werden. Das npm-Paket pouchdb-auth war jedoch leicht mit dem etwas älteren, nicht mehr unterstütztem pouchdb-authentication zu verwechseln. Lediglich das programmatische Erstellen von Nutzern im Docker Image konnte sich uns nicht erschließen. Zum Erstellen einer simplen Offline-First-Anwendung würden wir auch zukünftig wieder CouchDB verwenden, wenn ein objektbasiertes Datenmodell passend für den Use-Case ist.

Anfangs wurde der Pirate Bay Booking Client bloß mit jQuery¹⁰ und dem Materialize CSS Framework¹¹ umgesetzt. Dies war für den Anfang ein schneller Weg zur Umsetzung des ersten Interfaces, jedoch hat sich sehr schnell Spaghetticode entwickelt, da die verschiedenen Teilaufgaben der Anwendung nicht in verschiedenen Dateien gelagert war. Dort hat Svelte geholfen, da durch das Komponentensystem die verschiedenen Teilaufgaben einfach in verschiedene Komponenten behandelt werden konnten. Svelte war zudem noch sehr einfach zu lernen, da die Syntax auf HTML basiert, weshalb die Webapplikation neu implementiert und mit weiteren Features versehen werden konnte. In Zukunft würden wir wieder mit Svelte arbeiten, da es sehr wenig Code benötigt, um interaktive Interfaces zu erstellen und am Ende noch ein schlankes und performantes JavaScript Bundle erstellt.

¹⁰<https://jquery.com/>

¹¹<https://materializecss.com/>

A Benutzerhandbuch

Im folgenden Handbuch wird die Nutzung des Pirate Bay Tours Booking Clients erläutert. Die Funktionen der App werden umfangreich Schritt für Schritt vorgestellt. Angefangen mit dem Einloggen, über das Erstellen von Touren, sowie Hinzufügen neuer Reservierungen, bis zum Drucken der Liste mit Reservierungen. Bei der Vorstellung der Funktionen wird in einer logischen Reihenfolge vorgegangen, welche einen tatsächlichen Arbeitsablauf modellieren.

A.1 Starten der Anwendung und von CouchDB

Sowohl die Anwendung als auch CouchDB können per Mausklick direkt über die Docker-Anwendung gestartet werden, wie in folgendem Screenshot erkennbar.

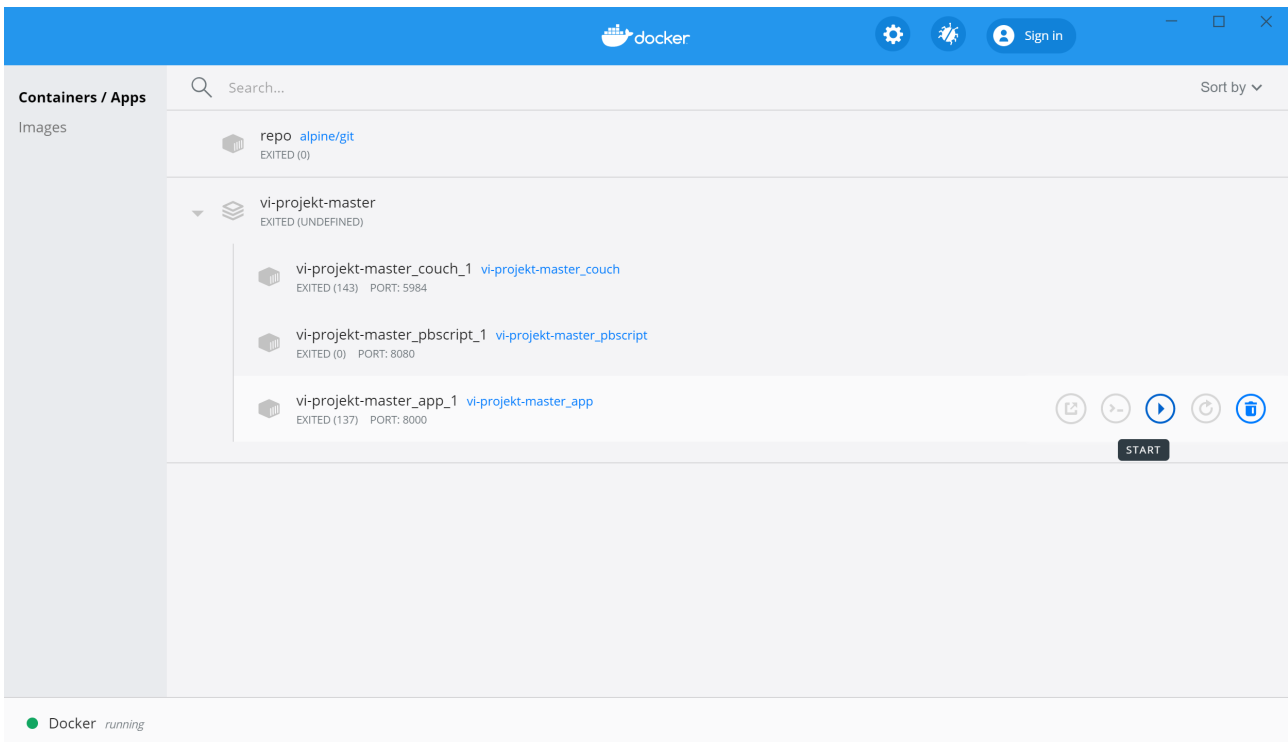
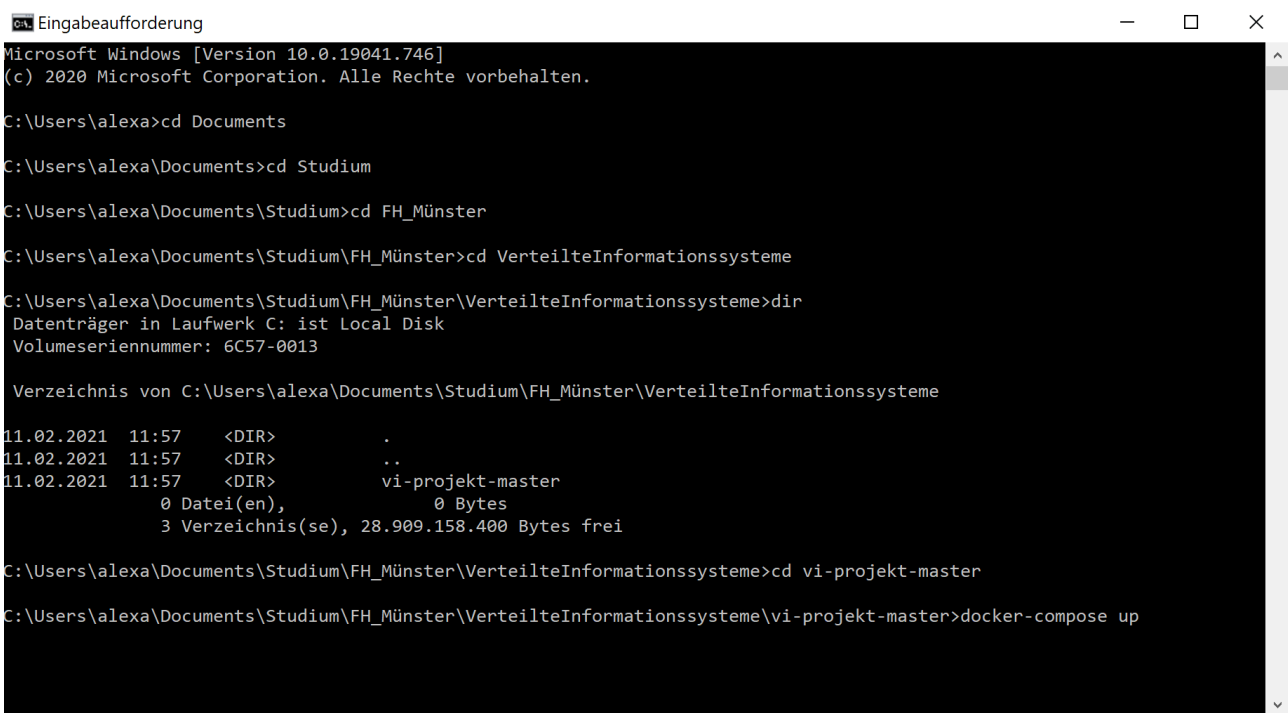


Abbildung 5: Starten der App und CouchDB über Docker Desktop

Auch über die Konsole ist der Start beider Komponenten möglich. Hierzu muss folgender Befehl, innerhalb des Anwendungsordners mit Installiertem Docker und docker-compose, in die Konsole eingegeben werden, wie im nächsten Screenshot erkennbar:

```
docker-compose up
```

```
Microsoft Windows [Version 10.0.19041.746]
(c) 2020 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\alexa>cd Documents

C:\Users\alexa\Documents>cd Studium

C:\Users\alexa\Documents\Studium>cd FH_Münster

C:\Users\alexa\Documents\Studium\FH_Münster>cd VerteilteInformationssysteme

C:\Users\alexa\Documents\Studium\FH_Münster\VerteilteInformationssysteme>dir
Datenträger in Laufwerk C: ist Local Disk
Volumenseriennummer: 6C57-0013

Verzeichnis von C:\Users\alexa\Documents\Studium\FH_Münster\VerteilteInformationssysteme
11.02.2021  11:57    <DIR>          .
11.02.2021  11:57    <DIR>          ..
11.02.2021  11:57    <DIR>          vi-projekt-master
               0 Datei(en),           0 Bytes
               3 Verzeichnis(se), 28.909.158.400 Bytes frei

C:\Users\alexa\Documents\Studium\FH_Münster\VerteilteInformationssysteme>cd vi-projekt-master

C:\Users\alexa\Documents\Studium\FH_Münster\VerteilteInformationssysteme\vi-projekt-master>docker-compose up
```

Abbildung 6: Starten der App und CouchDB über die Konsole

A.2 Einloggen

Die App ist erreichbar unter `localhost:8000/`, während man die CouchDB Weboberfläche unter `localhost:5984/_utils/` findet. Für die Nutzung der App ist eine Anmeldung erforderlich. Diese erfolgt über den „Log in“-Button, wie in der folgenden Abbildung zu erkennen.

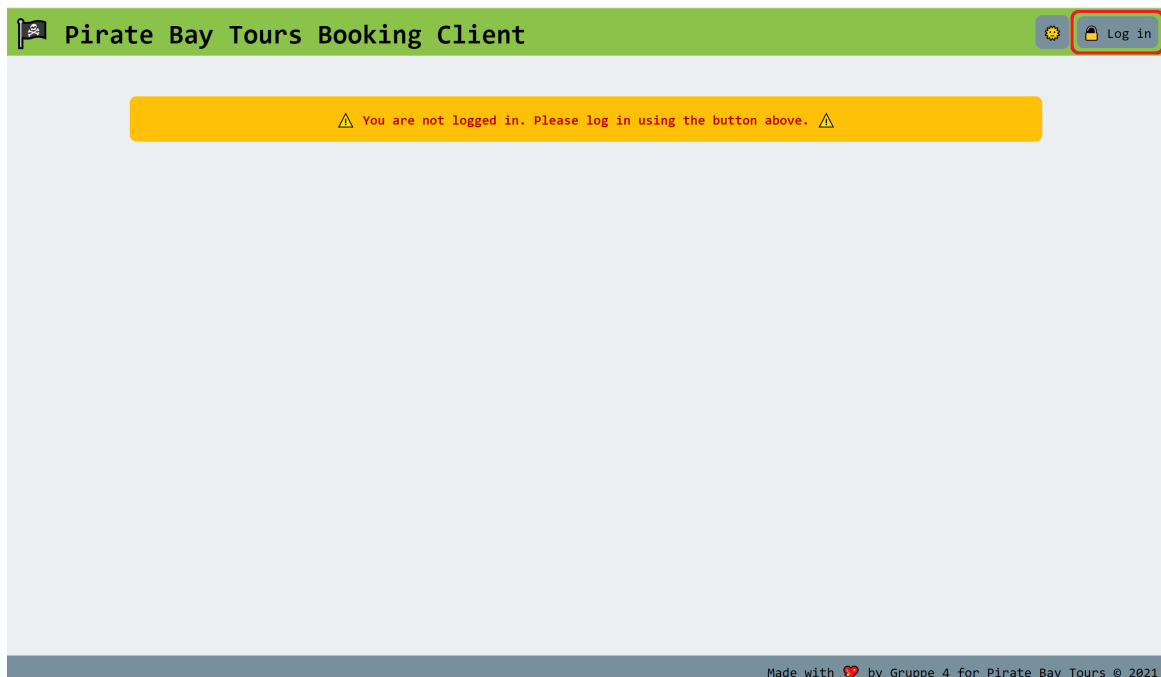


Abbildung 7: App vor Login



The image shows a modal window titled "Authentication" with a close button in the top right corner. It contains two input fields: "Username *" and "Password *". Below the fields is a green "Log in" button with a lock icon.

Abbildung 8: Login Fenster

Standardmäßig wird ein Admin Account mit dem Username admin und Passwort password angelegt. Dies sollte für eine Produktivumgebung umgehend über die CouchDB Weboberfläche geändert werden. Nach erfolgreichem Login kann der Prozess des Erstellens von Touren und Reservierungen beginnen.

A.3 Tour erstellen

Bevor der Agent beginnen kann Reservierungen hinzuzufügen, muss erst einmal die passende Tour erstellt werden. Zum Erstellen einer neuen Tour klickt man einfach auf den „add new tour“-Button, wie in der folgenden Grafik erkennbar.

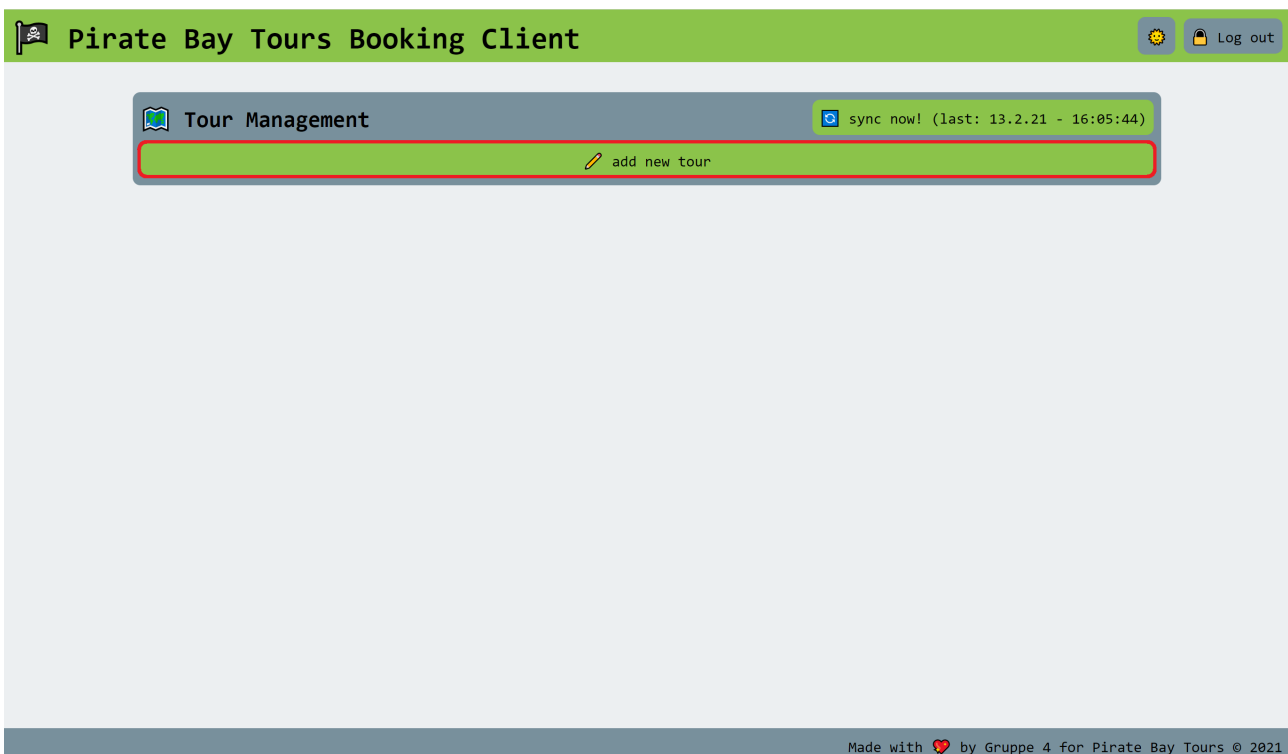
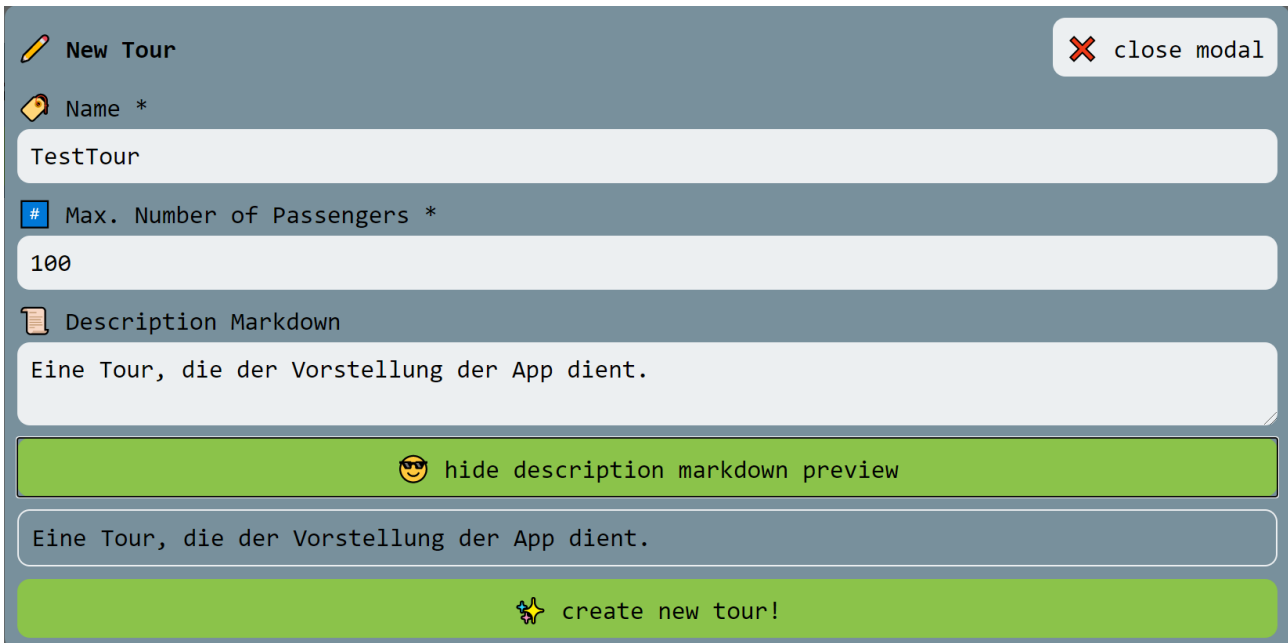


Abbildung 9: Tour erstellen

Im Anschluss öffnet sich ein neues Fenster, in dem die Details der Tour eingetragen werden

können. Zum endgültigen Anlegen der Tour müssen die Eingaben mit dem Klick auf den "create new tour"-Button bestätigt werden. Wie das aussieht, sieht man im folgenden Bild.



The image shows a modal window titled "New Tour" with a "close modal" button in the top right corner. The form contains the following fields and buttons:

- Name ***: A text input field containing "TestTour".
- Max. Number of Passengers ***: A number input field containing "100".
- Description Markdown**: A text area containing "Eine Tour, die der Vorstellung der App dient."
- hide description markdown preview**: A green button with a smiley face icon.
- create new tour!**: A green button with a star icon.

Below the "hide description markdown preview" button, a preview of the description is shown: "Eine Tour, die der Vorstellung der App dient."

Abbildung 10: Tour Erstellen Details

A.4 Sync-Button

An dieser Stelle soll der „sync now“-Button kurz erklärt werden. Sync steht in diesem Fall für Synchronisation. Das Betätigen des Buttons sorgt dafür, dass die Daten aus der App mit der ausgelagerten Datenbank synchronisiert werden. Daher kann der Button auch nur betätigt werden, wenn eine Verbindung zur Datenbank und damit eine aktive Internetverbindung vorliegt. Die Policy sieht vor, dass jeder Agent sowohl morgens zu Beginn der Arbeit und abends zum Schluss einmal den Button betätigt. Dadurch erhält der Agent jeden Morgen ein neues Kontingent an Buchungen, die er offline durchführen kann und jeden Abend werden seine Buchungen in die Datenbank geschrieben. Wo der Sync-Button zu finden ist, zeigt die folgende Abbildung. Der Button enthält immer den Zeitstempel der letzten Betätigung. Dies dient auch der eigenen Kontrolle, ob man den Sync schon durchgeführt hat.

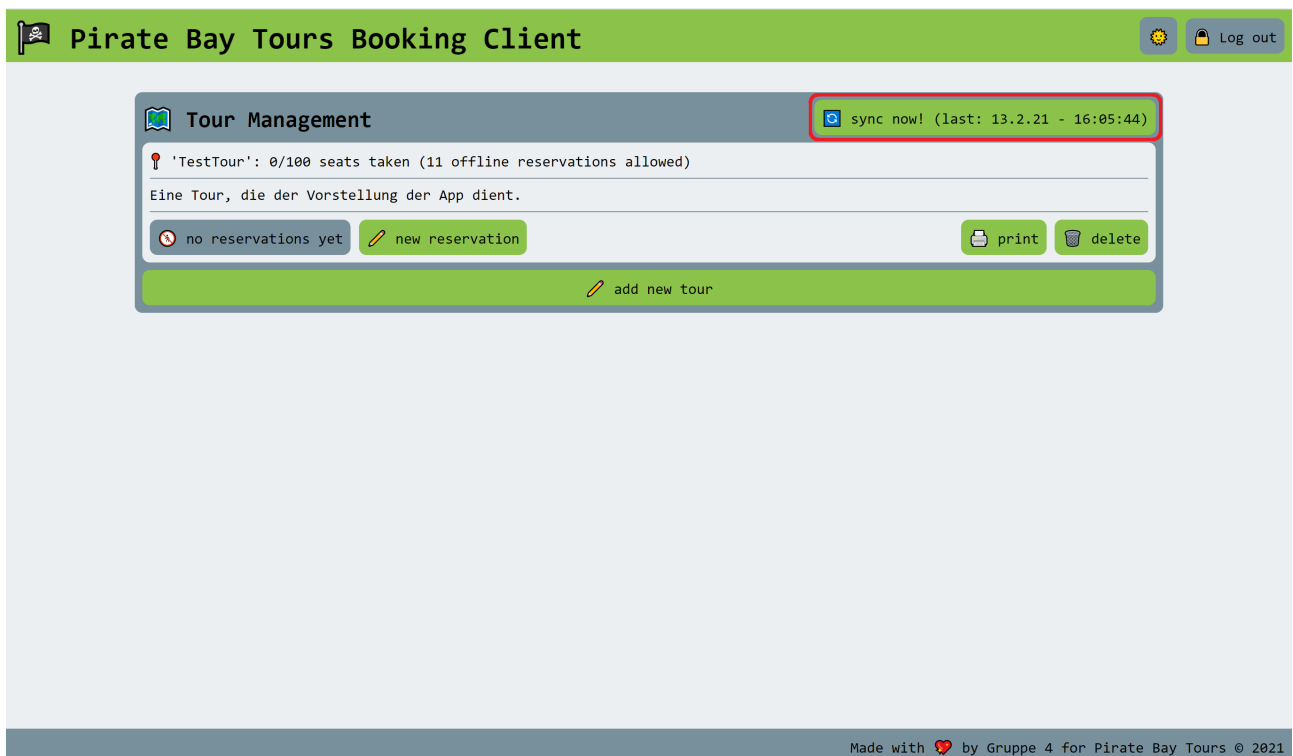


Abbildung 11: Der Sync-Button

A.5 Reservierungen hinzufügen

Wenn die Tour erfolgreich erstellt wurde, erscheint diese auch direkt in der App und es können Reservierungen hinzugefügt werden.

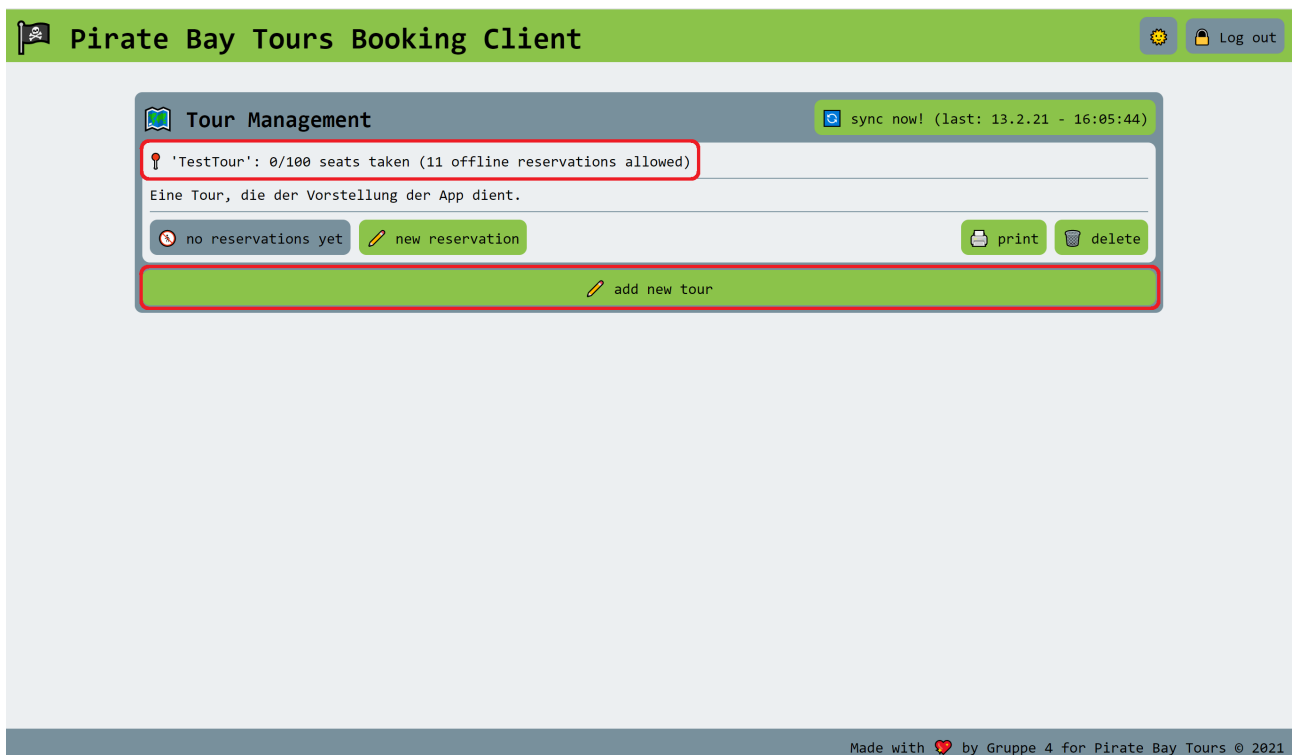


Abbildung 12: Überblick erstelle Tour

Für das Hinzufügen neuer Reservierungen sind allerdings einige Dinge zu beachten. Zum einen sieht man in der Abbildung 12 in rot markiert wie viele Sitze bereits belegt sind. Direkt rechts davon ist auch ersichtlich wie viele Reservierungen offline hinzugefügt werden können. Alle Reservierungen, die über diese Anzahl hinausgehen, können nicht sofort bestätigt werden. Hier muss erst eine Prüfung stattfinden. Möchte man nun eine neue Reservierung hinzufügen erscheint das folgende Fenster, um die Details einzugeben.



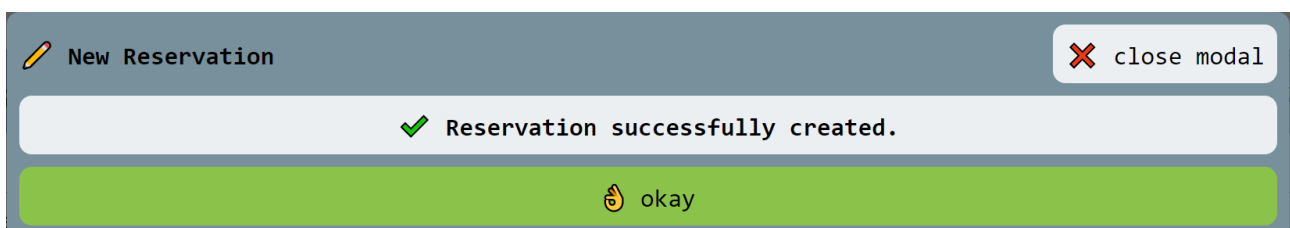
The screenshot shows a modal window titled "New Reservation" with a "close modal" button in the top right. The form contains the following fields:

- First name ***: Input field with "Max"
- Last name ***: Input field with "Mustermann"
- Phone number ***: Input field with "12345"
- Number of group members ***: Input field with "9"

At the bottom of the form is a green button with a star icon and the text "create new reservation!". The entire form area is highlighted with a red border.

Abbildung 13: Anlegen einer Reservierung

Wichtig ist es bei dem Hinzufügen der Reservierung auch die Kontaktdaten (Telefon) mitaufzunehmen. Dadurch kann der Buchende dann auch kontaktiert werden, wenn notwendig. Sollte die Anzahl der Gäste die maximale Anzahl an Offline-Buchungen nicht überschreiten, so wird nach Anlegen der Buchung folgende Meldung erscheinen.



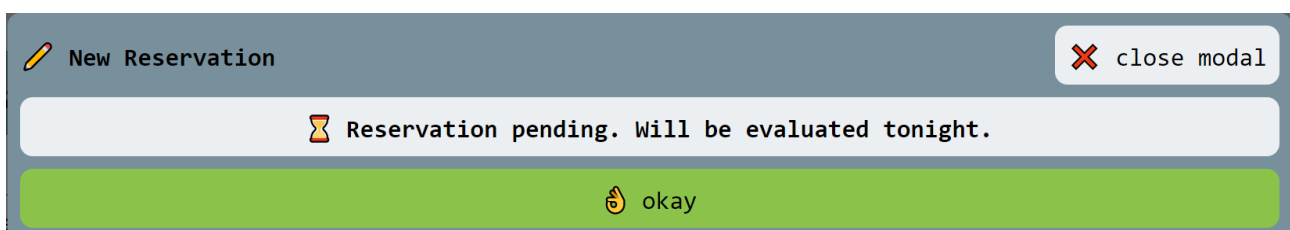
The screenshot shows the "New Reservation" modal with a success message displayed in a light blue box:

✓ **Reservation successfully created.**

Below the message is a green button with a hand icon and the text "okay".

Abbildung 14: Erfolgreiche Reservierung

Sollte die Anzahl allerdings die maximale Anzahl an Offline-Buchungen überschreiten, so kann die Buchung nicht sofort bestätigt werden. Die Meldung sieht dann wie folgt aus.



The screenshot shows the "New Reservation" modal with a pending message displayed in a light blue box:

⌚ **Reservation pending. Will be evaluated tonight.**

Below the message is a green button with a hand icon and the text "okay".

Abbildung 15: Reservierung muss geprüft werden

Wie der Text schon sagt, wird über Nacht ermittelt, ob die Buchung noch hinzugefügt werden kann oder nicht. Dies geschieht durch ein Skript, welches errechnet, ob noch genug Platz für die Gäste ist. Am nächsten Morgen kann dann der morgendliche Sync erfolgen, welcher einem verrät, ob die Gäste noch zur Tour zugelassen wurden. Über das Urteil wird man per Meldung in der App informiert.

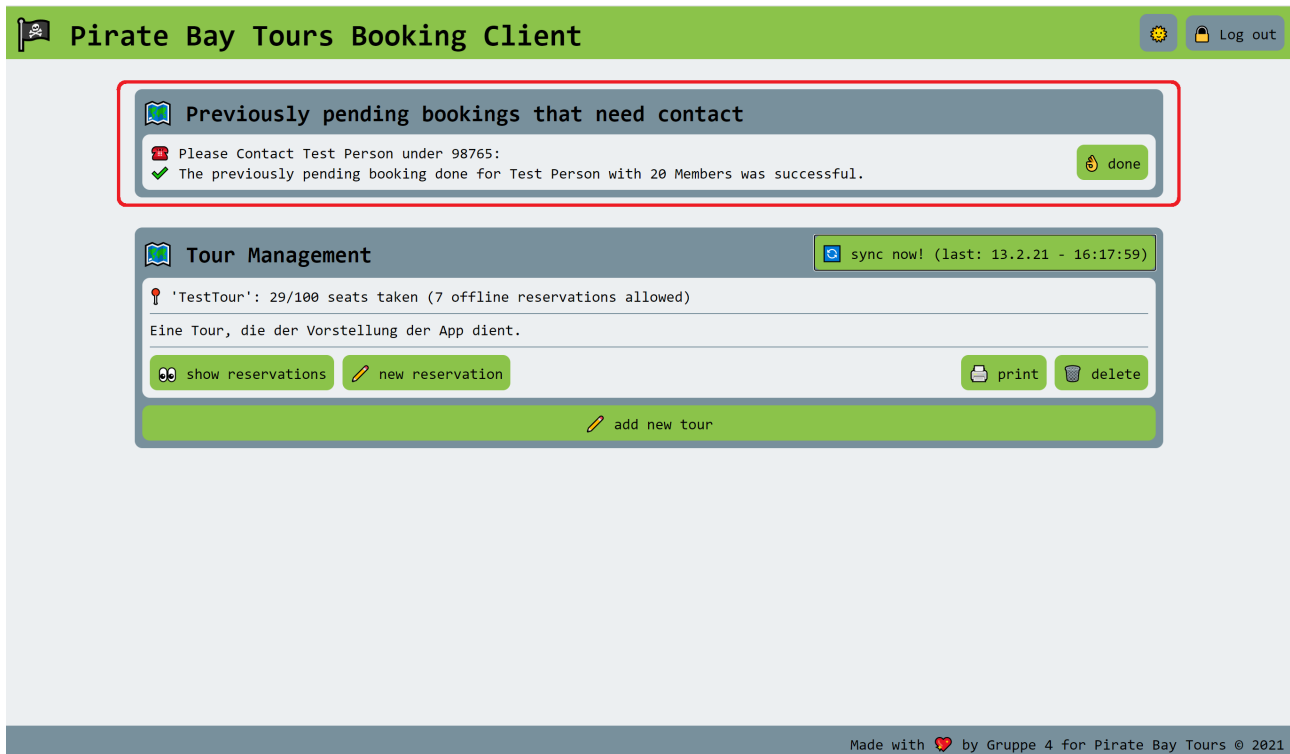


Abbildung 16: Geprüfte Buchung erfolgreich hinzugefügt

Jetzt sollte der Gast umgehend informiert werden, dass seine Reservierung erfolgreich war. Im Falle einer Ablehnung ist der Gast ebenfalls sofort zu informieren. Durch einen Klick auf den „done“-Button verschwindet die Meldung und die neue Reservierung wird auch unter den Reservierungen der Tour aufgeführt. Dies ist in Abbildung 17 zu sehen.

A.6 Löschen von Touren

Touren können bei falscher Anlage oder nach der Durchführung ganz einfach über den „delete“-Button gelöscht werden. Wenn man den Button betätigt, erscheint zuerst nochmal eine Abfrage, ob man sich sicher ist die entsprechende Tour löschen zu wollen.

A.7 Drucken der Liste der Reservierungen

Bevor die Tour startet, kann eine Liste mit den entsprechenden Reservierungen direkt über die Webanwendung ausgedruckt werden. Dadurch kann dann am Boot überwacht werden, ob auch alle Gäste erschienen sind. Um zu Drucken klickt man einfach auf den „print“-Button, im An-

schluss erscheint das Druckfenster des jeweiligen Browsers, wie in den folgenden Grafiken zu sehen.

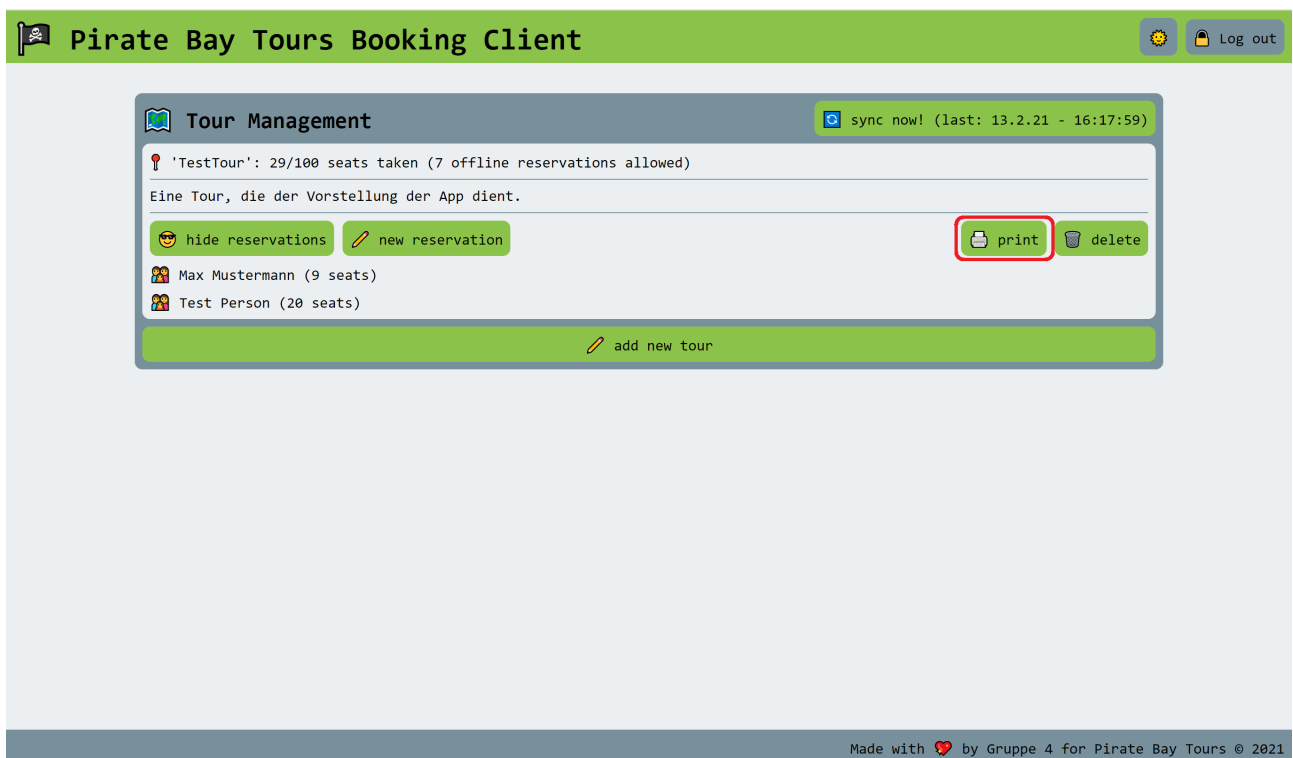


Abbildung 17: Druck ausführen

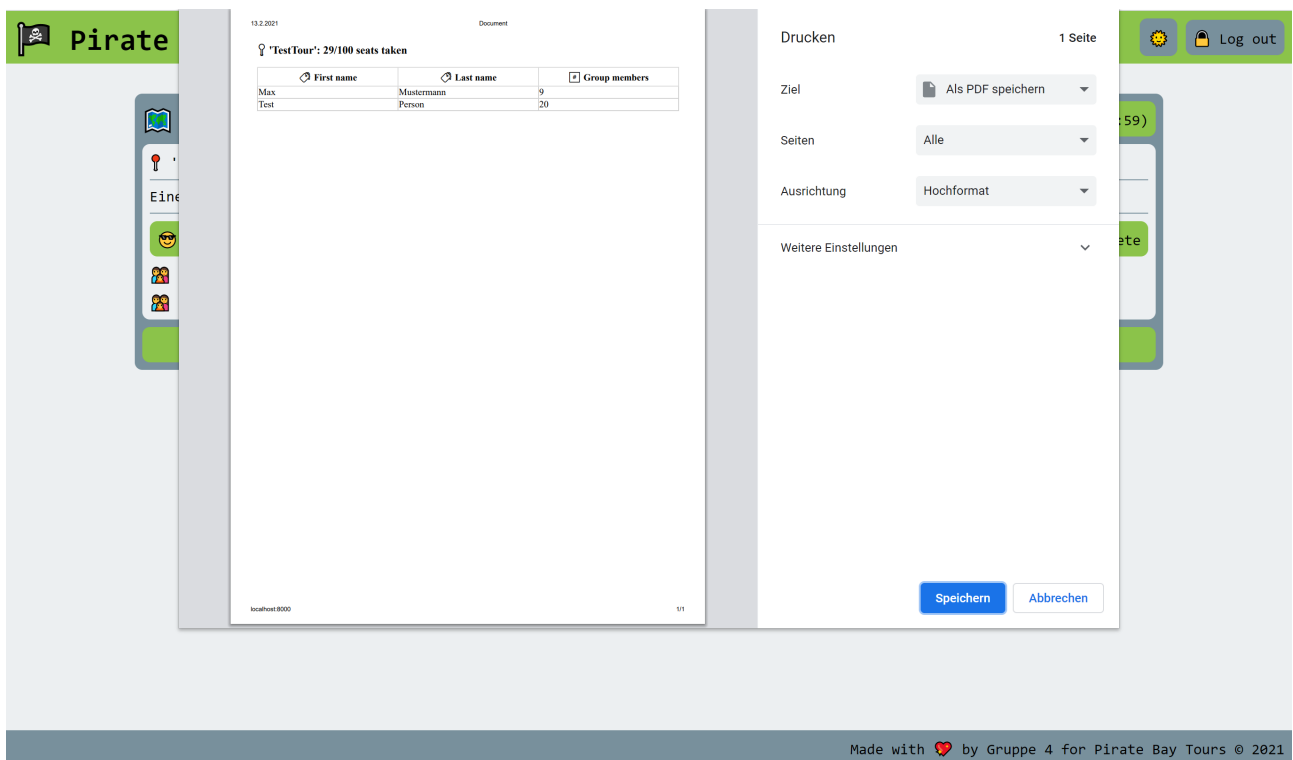


Abbildung 18: Druckeinstellungen anpassen

A.8 Extras

Jeder Agent kann seine App optisch gestalten, wie er/sie möchte. Dazu kann er/sie aus unterschiedlichen vorgefertigten Vorlagen wählen. Einfach auf das Logo neben dem „Log out“-Button klicken und auswählen.

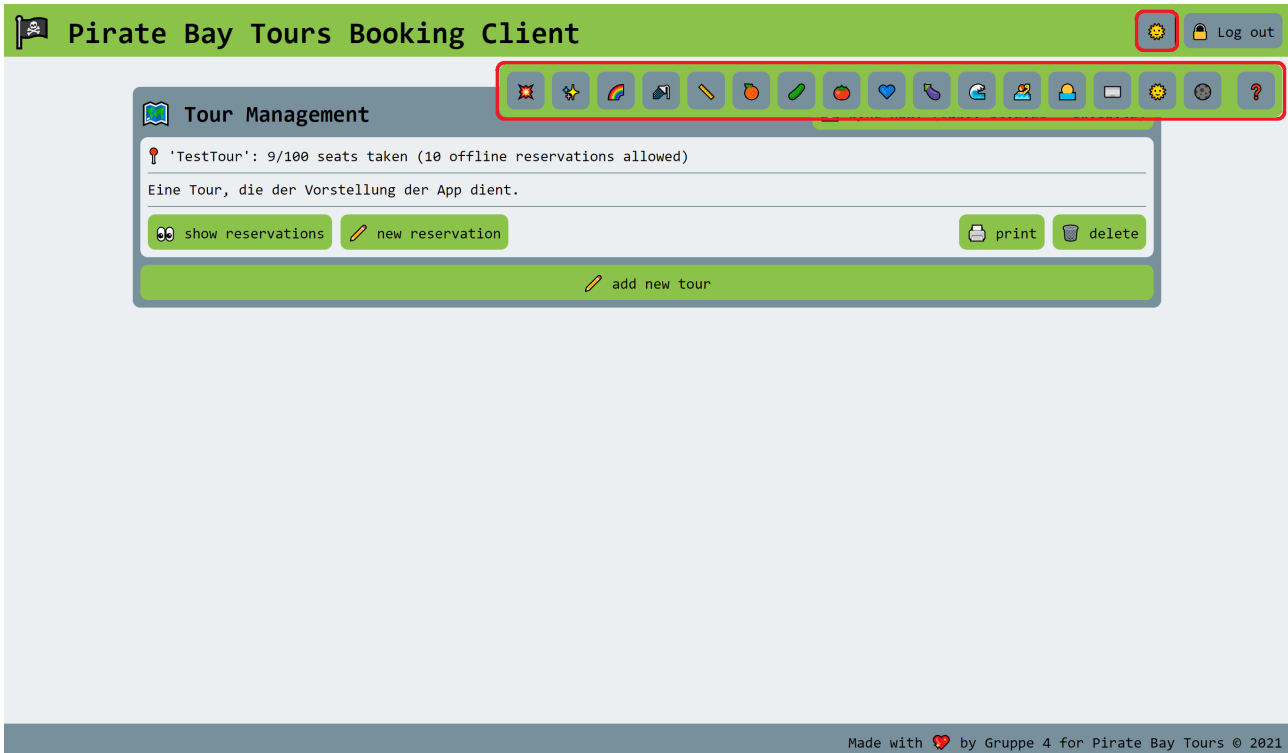


Abbildung 19: Neues Design wählen

Das neue Design könnte zum Beispiel so aussehen:

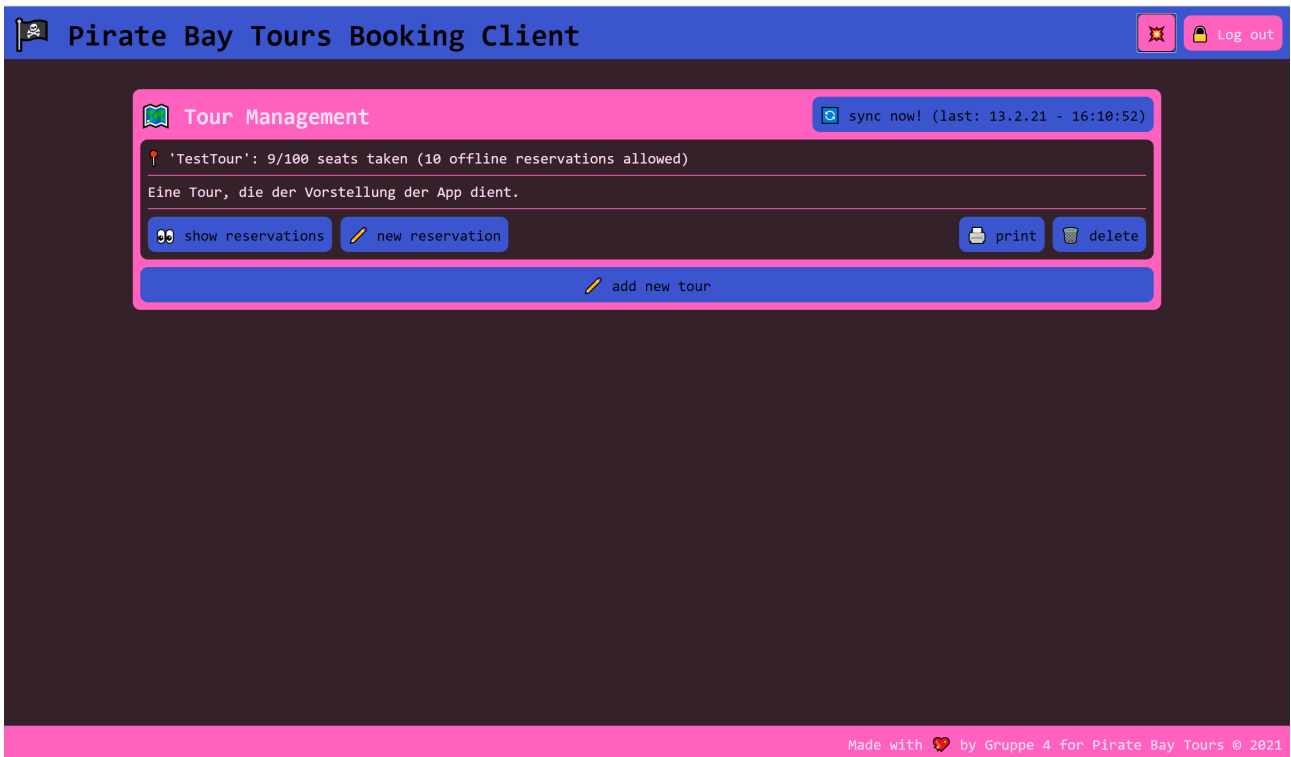


Abbildung 20: Neues Design