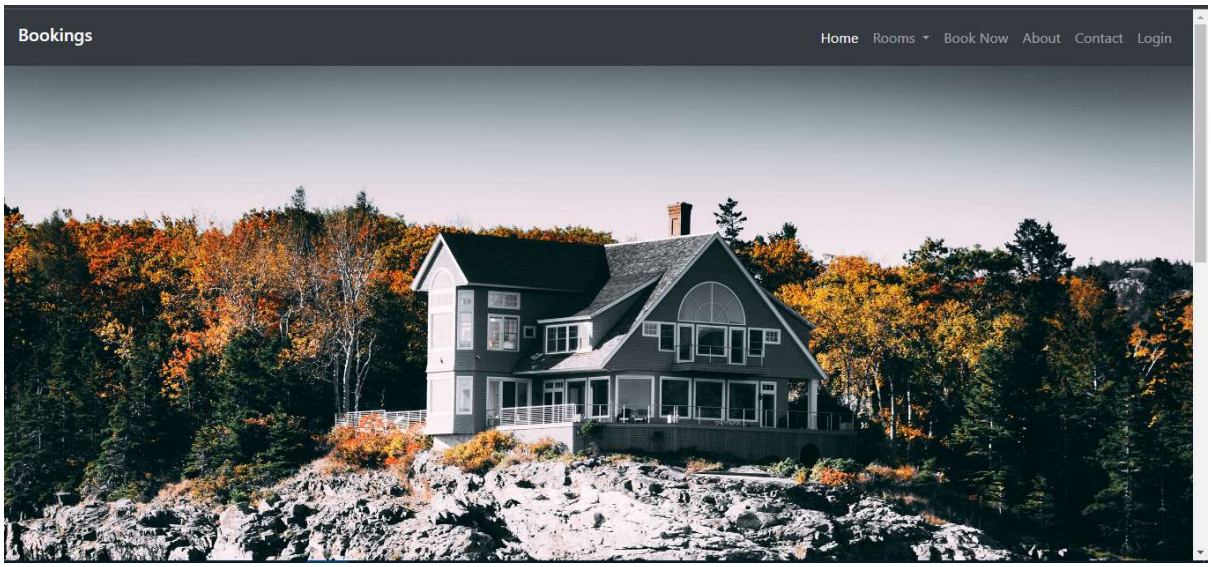

Hotelbuchungsprojekt – Dokumentation

Batuhan Yilmaz

Dipl.Wirtschaftsinformatiker HF

Inhalt

1. Aufgabenbeschreibung

Das Ziel dieses Projekts war die Entwicklung eines **Hotelbuchungssystems** mit einem Go-Backend und einem Web-Frontend. Das System ermöglicht es Nutzern, Zimmer zu buchen, die Verfügbarkeit von Zimmern zu überprüfen und Details über die angebotenen Zimmer zu sehen. Zusätzlich gibt es ein **Administrator-Dashboard**, das es Admins ermöglicht, Reservierungen zu verwalten und den Buchungskalender einzusehen.

2. Erfolgskriterien

Backend: Implementierung eines Go-Backends, das Daten per REST-API verarbeitet und speichert. Die Daten bleiben auch nach Neustarts der Applikation in einer PostgreSQL-Datenbank erhalten.

Frontend: Entwicklung eines benutzerfreundlichen Web-Frontends, das die Services des Backends konsumiert und Nutzern die Möglichkeit gibt, Buchungen vorzunehmen. Es wurde mit HTML, CSS und JavaScript umgesetzt.

Admin-Dashboard: Eine Verwaltungsoberfläche für Administratoren, mit der sie Buchungen anzeigen, verwalten und den Belegungsstatus der Zimmer in einem Kalender einsehen können.

Sicherheit: Integration von Sicherheitsmassnahmen wie **CSRF-Schutz** und **Session-Management** zur Absicherung der Web-Anwendung.

3. Vorgehen

Projektplanung

Zunächst wurde das Konzept des Hotelbuchungssystems erstellt. Dabei wurden folgende Technologien und Tools festgelegt:

- **Backend: Go**
- **Frontend: HTML, CSS, JavaScript**
- **Datenbank: PostgreSQL**
- **Bibliotheken:**
- **«go-chi» für Routing,**
- **«pgx» für die PostgreSQL-Anbindung,**
- **«scs» für Session-Management,**
- **«nosurf» für CSRF-Schutz.**

Implementierung

Backend:

In der Datei «**main.go**» wird der Einstiegspunkt der Applikation definiert. Hier wird die Verbindung zur PostgreSQL-Datenbank hergestellt, Sessions werden verwaltet und die Routen für die API-Anfragen werden über das «**go-chi**»-Routing erstellt.

- Es gibt eine Middleware-Schicht, die Sicherheitsfunktionen wie CSRF-Schutz und Session-Handling implementiert.
- Der E-Mail-Versand wurde implementiert, um den Nutzern eine Bestätigung ihrer Buchung per E-Mail zukommen zu lassen.

Frontend:

Die Seiten wurden mit Go HTML-Templates erstellt. Diese Templates generieren dynamisch HTML-Seiten, wie z. B. die Startseite («**home.page.gohtml**») und die Buchungsübersicht («**reservation-summery.page.gohtml**»).

- Die UI-Elemente sind benutzerfreundlich gestaltet, um den Buchungsprozess so einfach wie möglich zu gestalten.

Admin-Dashboard:

Das Dashboard ermöglicht es Administratoren, über Seiten wie «**admin-dashboard.page.gohtml**» und «**admin-reservations-calendar.page.gohtml**» den Buchungsstatus und die Belegung der Zimmer zu sehen und Reservierungen zu verwalten.

4. 2-3 Probleme und Lösungsbeschreibung

1. Problem: Datenbankverbindung

- Beschreibung: Während der Implementierung kam es zu Verbindungsproblemen mit der PostgreSQL-Datenbank. Dies lag an falsch konfigurierten Verbindungsparametern.
- Lösung: Nach sorgfältiger Überprüfung der Umgebungsvariablen für die Verbindung wurde das Problem gelöst, indem die korrekten Parameter in der `run.sh`-Datei verwendet wurden.

2. Problem: CSRF-Schutz

- Beschreibung: Zu Beginn funktionierte der CSRF-Schutz (`nosurf`) nicht korrekt und blockierte einige POST-Anfragen.
- Lösung: Das Problem wurde behoben, indem der CSRF-Token in die HTML-Formulare eingebettet wurde. Die Middleware wurde korrekt eingebunden, sodass POST-Anfragen sicher verarbeitet werden konnten.

3. Problem: Template-Caching im Produktionsmodus

Beschreibung: Im Entwicklungsmodus wurden Änderungen an den HTML-Templates nicht sofort reflektiert.

- Lösung: Ein Cache-Flag (`useCache`) wurde in der Anwendung eingeführt. Während der Entwicklung wird das Caching deaktiviert, um Änderungen an den Templates sofort sichtbar zu machen.

5. Lessons Learned

Go für Webentwicklung: Go eignet sich hervorragend für die Entwicklung von Webanwendungen. Es bietet eine starke Performance und gute Werkzeuge für Routing, Middleware und Datenbankanbindungen.

Session-Management: Die Verwendung von `scs` für die Session-Verwaltung erwies sich als robust und zuverlässig, insbesondere für die Authentifizierung von Benutzern.

Middleware und Sicherheit: Die Implementierung von CSRF-Schutz und Sessions war zunächst herausfordernd, hat aber gezeigt, wie wichtig Sicherheitsmassnahmen bei der Entwicklung von Webanwendungen sind.

Templates und Caching: Der Umgang mit Go HTML-Templates war lehrreich, insbesondere beim Entwickeln eines Caching-Mechanismus für den Produktionsmodus.

6. Qualität des Programmcodes

Der Code wurde durch Tests und Logging kontinuierlich überprüft. Es wurde sichergestellt, dass das System effizient arbeitet, und Fehler wurden durch detaillierte Log-Nachrichten schnell gefunden und behoben.