ÜK223 Dokumentation: Raumreservierungs-System

13.12.2024 Levin, Dani, Matheus

1. Einführung

Das Raumreservierungs-System ist eine Webanwendung, die entwickelt wurde, um Buchungsprozesse für Räume zu vereinfachen. Es erlaubt Benutzern, Reservierungen mit spezifischen Details wie Zeit, Ort, Bemerkungen und Teilnehmern zu erstellen, zu bearbeiten, anzuzeigen und zu löschen. Das System setzt auf ASP.NET Core und SQLite und bietet robuste Sicherheits- und Validierungsmechanismen.

2. Hauptfunktionen

2.1 Erstellen von Reservierungen

- Benutzer können über eine benutzerfreundliche Weboberfläche neue Buchungen erstellen.

- Validierungen verhindern Überschneidungen mit bestehenden Reservierungen.

2.2 Bearbeiten von Reservierungen

- Reservierungen können bei Bedarf geändert werden, z. B. Teilnehmerlisten oder Zeiträume.

- Der Zugriff auf diese Funktion erfolgt über private Schlüssel.

2.3 Anzeigen von Reservierungen

- Buchungen können öffentlich (nur lesbar) oder privat (voller Zugriff) angezeigt werden.

- Öffentliche Ansichten sind auf grundlegende Details beschränkt.

2.4 Löschen von Reservierungen

- Benutzer können bestehende Reservierungen vollständig löschen.

- Sicherheitsabfragen verhindern versehentliches Löschen.

2.5 Validierungsmechanismen

- Eingaben werden sowohl auf Client- als auch auf Serverseite validiert.

- Regeln stellen sicher, dass Teilnehmer, Bemerkungen und Zeiträume korrekt sind.

3. Systemarchitektur

3.1 MVC-Architektur

Das Projekt folgt dem Model-View-Controller (MVC)-Ansatz:

- \*\*Model\*\*: Definiert Datenstrukturen wie die Klasse `Reservation`.

- \*\*View\*\*: HTML-Seiten zur Darstellung von Daten (z. B. `Index.html`, `Create.html`).

- \*\*Controller\*\*: Steuert die Interaktion zwischen Model und View (z. B. `HomeController.cs`, `ReservationController.cs`).

3.2 Datenbankstruktur

- Datenbank: SQLite.

- Tabellenstruktur:

- \*\*Reservations\*\*: Speichert Informationen wie Datum, Zeit, Raum, Bemerkungen und Teilnehmer.

- Migrations wie `InitialCreate` und `UpdateModel` definieren die Datenbankstruktur.

3.3 Sicherheitsmaßnahmen

- Schüsselbasierter Zugriff:

- \*\*Privater Schlüssel\*\*: Ermöglicht Bearbeitung und Löschung.

- \*\*Öffentlicher Schlüssel\*\*: Beschränkt auf lesenden Zugriff.

- Validierung von Eingaben auf Client- und Serverseite.

4. Code-Struktur

4.1 Beispiel: `ReservationController.cs`

public class ReservationController : Controller

{

public IActionResult Create()

{

return View();

}

[HttpPost]

public IActionResult Create(Reservation reservation)

{

if (ModelState.IsValid)

{

\_context.Reservations.Add(reservation);

\_context.SaveChanges();

return RedirectToAction("Success");

}

return View(reservation);

}

}

4.2 Beispiel: `Index.html`

<!DOCTYPE html>

<html lang="en">

<head>

<title>Raumreservierung</title>

</head>

<body>

<h1>Willkommen zur Raumreservierung</h1>

<a href="/Reservation/Create">Neue Reservierung erstellen</a>

</body>

</html>

4.3 Beispiel: `ApplicationDbContext.cs`

public class ApplicationDbContext : DbContext

{

public DbSet<Reservation> Reservations { get; set; }

}

5. Installation und Konfiguration

5.1 Voraussetzungen

- Webserver mit ASP.NET Core-Unterstützung.

- SQLite-Datenbank.

5.2 Schritte

1. Projekt klonen:

git clone <repository-url>

2. Abhängigkeiten installieren:

dotnet restore

3. Datenbank migrieren:

dotnet ef database update

4. Projekt starten:

dotnet run

6. Fazit

Das Raumreservierungs-System bietet eine umfassende Lösung für die Verwaltung von Raumbuchungen. Mit einem Fokus auf Benutzerfreundlichkeit, Sicherheit und Funktionalität ist es ideal für Organisationen, die eine effiziente Raumbuchungsplattform benötigen.