住民管理システム プロジェクト仕様書

基本設計

要件定義

ルート詳細

開発分担案

テスト

HIDESHIMA

システム基本設計（使用技術）

SERVER SIDE

C# / ASP.NET Core

ビジネスロジックの構築、データベース連携、API エンドポイントの制御を担う堅牢なサーバー基盤として採用しています。

CLIENT SIDE

Frontend Stack

シンプルかつ高速なユーザー体験を提供するため、モダンなUIフレームワークと標準技術を組み合わせています。

HTML 5	CSS 3	JavaScript
Bootstrap 5	Vanilla CSS	Vanilla JS

要件定義

登録に必要な列名

名前	物理名	型
住民ID	Id	int
姓名	LastName / FirstName	nvarchar
ふりがな	LastNameKana / FirstNameKana	nvarchar
住所情報	PostalCode / Address / HouseNumber / BuildingName	nvarchar
属性	DateOfBirth / Gender	datetime / nvarchar
連絡先等	Email / PhoneNumber / Notes	nvarchar
登録日時	CreatedAt	datetime

ルート設計

- `/Home/Index` - HOME(TOP)画面
- `/Resident/Index` - 住民一覧表示・検索
- `/Resident/Details/{id}` - 住居情報詳細
- `/Resident/Create` - 住民登録
- `/Resident/Edit/{id}` - 住情報更新
- `/Resident/Delete/{id}` - 住民削除処理

将来的な拡張性

開発段階：SQLLite
運用段階：SQLerver等の大規模RDBMSへコードの変更
最小限に移行可能なEntity Framework Coreを採用。

管理者用ページ (Admin)

一般ユーザーとは別に管理機能を設ける。
特定のロールを持つユーザーのみがアクセス可能なページとして実装。
住民情報の一括管理や権限設定を行う。

ルート詳細

機能名	メソッド	パス	コントローラー / ファイル	ビューファイル	詳細説明
TOPページ	GET	/Home/Index	Controllers/HomeController.cs	Views/Home/Index.cshtml	システムの入り口。新規登録と一覧へのナビゲーションを提供。
住民一覧	GET	/Resident/Index	Controllers/ResidentController.cs	Views/Resident/Index.cshtml	全住民情報を取得し、一覧を表示。条件による絞り込み検索が可能。
詳細表示	GET	/Resident/Details/{id}	Controllers/ResidentController.cs	Views/Resident/Details.cshtml	対象住民の全情報を美しくレイアウトして表示。
新規登録（画面）	GET	/Resident/Create	Controllers/ResidentController.cs	Views/Resident/Create.cshtml	登録用フォーム画面を表示。
新規登録（処理）	POST	/Resident/Create	Controllers/ResidentController.cs	-	データを検証し、DBに保存。
編集（画面）	GET	/Resident/Edit/{id}	Controllers/ResidentController.cs	Views/Resident/Edit.cshtml	指定IDの情報を取得し、編集画面を表示。
編集（処理）	POST	/Resident/Edit/{id}	Controllers/ResidentController.cs	-	変更を検証し、レコードを更新。
削除確認（画面）	GET	/Resident/Delete/{id}	Controllers/ResidentController.cs	Views/Resident/Delete.cshtml	削除対象を表示し、確認を求める。
削除（処理）	POST	/Resident/DeleteConfirmed	Controllers/ResidentController.cs	-	DBからレコードを物理削除。

開発分担案 (大規模開発向け)

規模開発において並行作業を効率化するための推奨分担案です。

1. インフラ・共通基盤チーム

Core

データベースサーバーの構築、認証・認可の共通機能実装、APIの共通エラーハンドリングなど、アプリケーションの土台を担当します。

2. バックエンド (API/ロジック) チーム

Server-side

C#を使用したビジネスロジックの実装、データベースのI/O処理、外部システムとの連携APIの開発を担当します。

3. フロントエンド (UI/UX) チーム

Client-side

HTML/CSS/JSを使用した画面の構築、ユーザービリティの向上、バックエンドAPIとのデータ連携実装を担当します。

4. 品質管理 (QA) ・テストチーム

Testing

単体テスト・結合テストのシナリオ作成、自動テストスクリプトの作成、バグの検出および進捗管理を担当します。

5. デザイン・ドキュメントチーム

Design & Docs

UIプロトタイプの作成、アイコンの制作、仕様書・マニュアルの整備。本ファイルのメンテナンスもここに含まれます。

テスト

デバックリスト	チェック欄
TOP画面遷移テスト HOME画面が正しく表示され、各機能へのリンクが有効か。	未 OK 修正
初期表示テスト 一覧画面において、登録済みの住民データが正しく読み込まれるか。	未 OK 修正
バリデーションテスト 必須項目が空の状態で「登録」を押下した際、警告が表示されるか。	未 OK 修正
保存処理テスト 新規登録が成功し、データベースにレコードが作成されるか。	未 OK 修正
編集/削除テスト 特定のデータの編集、および削除が正常に行えるか。	未 OK 修正