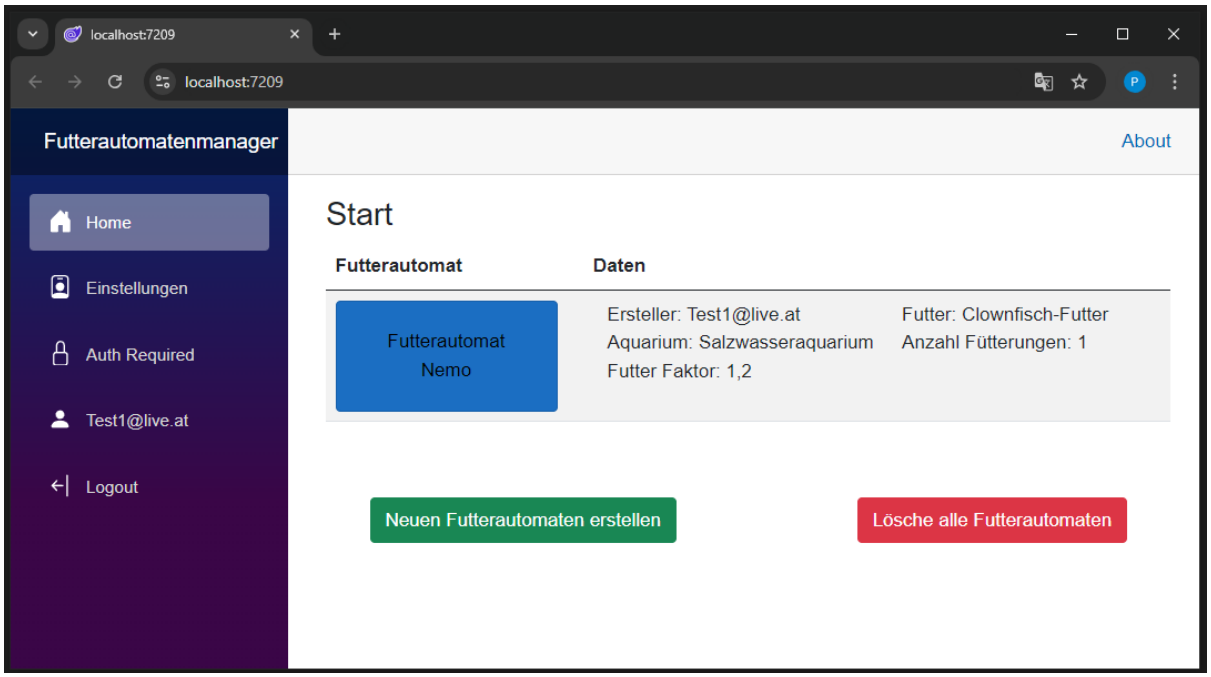
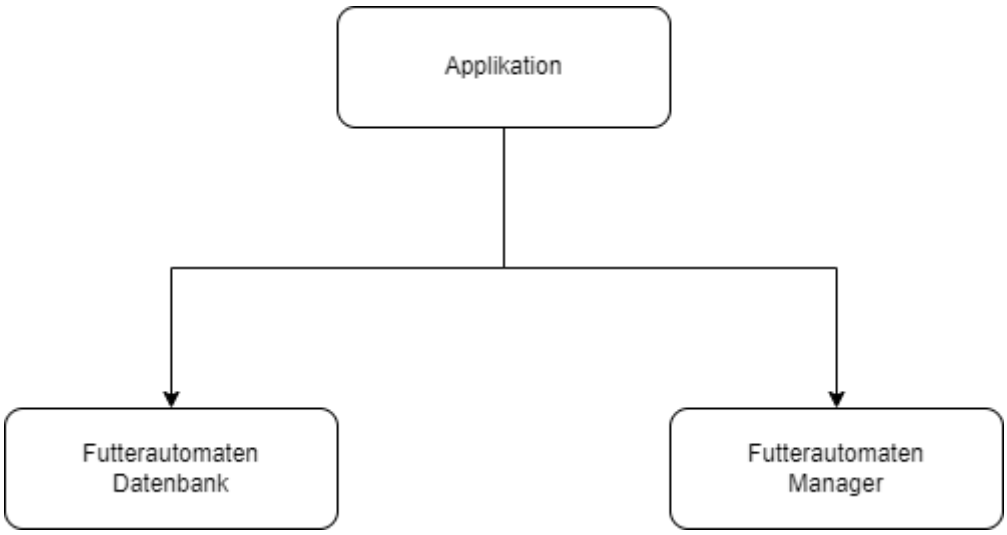
Futterautomatenmanager
Technische Dokumentation



1 Aufbau der Applikation “Futterautomatenmanager”

Zum Ausführen der Applikation dient die Datei Futterautomatenmanager.exe



Die Applikation besteht aus 2 Projekten.

- Der Futterautomaten-Datenbank:
  Diese beinhaltet alle Entitäten, welche für die Futterautomaten von Relevanz sind.
- Dem Futterautomaten-Manager
  Dieser ist für die Interaktion mit dem User zuständig und gibt die eingestellten Parameter an die Datenbank weiter.

1.1 FutterautomatenDatenbank

namespace FutterautomatenDatenbank.Models

In diesem befinden sich die folgenden Entitäten:

- Aquarium.cs
- Fuetterungen.cs
- Futter.cs
- Futterautomat.cs
- Person .cs

Ein Interface:

- IFutterautomatenEFCoreRepoository.cs

Und die Klasse:

- FutterautomatenEFCoreRepository.cs

Diese Klasse implementiert das Interface IFutterautomatenEFCoreRepoository.cs und ist auf mehrere Partielle Klassen aufgeteilt:

- FutterautomatenEFCoreRepositoryPartialAquarium.cs
- FutterautomatenEFCoreRepositoryPartialFuetterung.cs
- FutterautomatenEFCoreRepositoryPartialFutter.cs
- FutterautomatenEFCoreRepositoryPartialPerson.cs

Die Klasse FutterautomatenEFCoreRepository.cs enthällt alle notwendigen Operationen zur Interaktion mit der Datenbank.
Beispiele hierfür sind speichern, updaten und löschen von Datenbankeinträgen.

namespace FutterautomatenDatenbank.Context

Hier befindet sich die Klasse FutterautomatenContext.cs.
Diese wiederum erbt von DbContext und beinhaltet alle Listen / Tabellen der Datenbank.
Der Konstruktor übernimmt ein Objekt von der Klasse DbContextOptions<FutterautomatenContext> und übergibt dieses an seine Basisklasse.
Dieses Objekt wird durch Dependency injection aus dem Projekt Futterautomatenmanager bei der Erzeugung des Cotext zur Verfügung gestellt.

1.2 FutterautomatenManager

namespace Futterautomatenmanager.Components

- Account:
  Beinhaltet die Klassen und Razor Dateien für die Authentifizierung.
- Controls:
  Enthällt die wiederverwendbaren Komponenten der Applikation.
  Beispiel:
  FutterautomatComponent.razor kann zur Laufzeit verwendet werden, um neue Futterautomaten im Userinterface darzustellen. Dieses Element besteht wiederum aus einer FutterautomatButtonComponent.razor und einer FutterautomatStatsComponent.razor Komponente.
- Erweiterungsmethoden
- Layout:
  Hierin befindet sich das Mainlayout und die Navigationsleiste der Anwendung
- Pages:
  Umfasst alle Seiten, die mittels URI angesteuert werden können.
  - Settings:
    - Aquariumverwaltung.razor
    - Futterverwaltung.razor
  - Auth.razor
  - Einstellungen.razor
  - Error.razor
  - NeueFuetterung.razor
  - NeuerFutterautomat.razor
  - NeuesAquarium.razor
  - NeuesFutter.razor
  - Start.razor

namespace Futterautomatenmanager.Controllers

- FutterautomatController.cs
  Dies ist der API-Controller für die Applikation.
  Mithilfe von [HttpGet] bzw. [HttpGet("{bezeichnungFutterautomat}")] können über den Browser die Futterautomaten abgefragt werden.

namespace Futterautomatenmanager.Data

Stellt die Klassen für die Authentifizierung zur Verfügung.

namespace Futterautomatenmanager.Models.APIModels

Bietet Helferklassen für die Ausgabe der Futterautomaten über die API

Aktivitätsdiagramm Futterautomat erstellen