DOCUMENTAITE DIERENTUIN

[Document subtitle]

Denzel Bendt

Ebram Moawad

Mitchel Meskes

CONTENTS

INLEIDING

De eindopdracht voor het vak 'Software Language 2b' gaat over het maken en ontwikkelen van een werkend dierentuinbeheersysteem. Dit project geeft ons een geweldige kans om de kennis en vaardigheden die we tijdens de lessen hebben geleerd, in de praktijk te brengen. Het belangrijkste doel is om een webapplicatie en een API te maken waarmee gebruikers verschillende dingen in een virtuele dierentuin kunnen beheren, zoals dieren, categorieën en verblijven.

In de wereld van softwareontwikkeling is het belangrijk dat je niet alleen kunt programmeren, maar ook begrijpt hoe verschillende onderdelen van een systeem samenwerken en hoe je ze goed kunt integreren. Deze opdracht helpt ons om die samenwerking en integratie te leren. Het zorgt ervoor dat we een stevige basis krijgen voor het ontwikkelen van sterke en goed werkende softwaretoepassingen.

WAT IS HET VOOR APPLICATIE

Het dierentuinbeheersysteem is een website waarmee gebruikers verschillende onderdelen van een dierentuin kunnen beheren. Met dit systeem kunnen gebruikers dieren aanmaken, bekijken, bewerken en verwijderen. Ze kunnen ook categorieën beheren waarin de dieren zijn ingedeeld en de verblijven beheren waar de dieren wonen.

Naast deze basisfuncties heeft het systeem ook extra mogelijkheden, zoals het simuleren van zonsopgang en zonsondergang, het regelen van voedertijden en het automatisch toewijzen van dieren aan verblijven.

PLAN VAN AANPAK

Project: Samenwerking van Groep 6 en Groep 13

---

1. Doelstelling

Het doel van dit project is om een robuust en functioneel systeem te ontwikkelen dat voldoet aan de gestelde eisen. We willen een gebruiksvriendelijke database creëren met een goed functionerende backend en een intuïtieve gebruikersinterface.

2. Taken en Verantwoordelijkheden

- **Ebram**
    - Migraties
    - Controllers
    - Seeding van de database
    - Backend-logica
    - Schrijven van unit tests
- **Mitchel**
    - Opzetten en optimaliseren van de database
    - Implementeren van de gebruikersinterface (UI)
    - Unittests schrijven en uitvoeren
    - Documentatie voltooien
- **Denzel**
    - Documentatie schrijven
    - Feedback verzamelen
    - Gebruikerssupport en feedbacksysteem opzetten
    - Trainingsmateriaal voorbereiden

3. Projectfases

Fase 1: Voorbereiding en Planning

- Samenvoegen van de repositories van groep 6 en groep 13.
- Verdeling van taken en verantwoordelijkheden.
- Opstellen van een gedetailleerd plan van aanpak.

Fase 2: Basisopzet

- Ebram zet de basisstructuur van het project op met migraties en controllers.
- Mitchel richt zich op de opzet van de database en de gebruikersinterface.

- Denzel begint met het schrijven van de documentatie en verzamelt eerste feedback.

Fase 3: Ontwikkeling

- Ebram werkt verder aan de backend-logica en voert unit tests uit.
- Mitchel verbetert de gebruikersinterface en voert unittests uit om de functionaliteit te waarborgen.
- Denzel blijft feedback verzamelen en werkt aan de documentatie en gebruikerssupport.

Fase 4: Integratie en Testen

- Samenvoegen van de verschillende onderdelen van het project.
- Uitvoeren van uitgebreide tests om bugs te identificeren en op te lossen.
- Optimaliseren van de prestaties en gebruikerservaring.

Fase 5: Oplevering en Evaluatie

- Voltooiing van de documentatie en het trainingsmateriaal.
- Organiseren van een trainingssessie voor gebruikers.
- Evaluatie van het project en identificatie van verbeterpunten.

4. Tijdschema

- **Week 1-2:** Voorbereiding en planning
- **Week 3-4:** Basisopzet en eerste ontwikkelingsstappen
- **Week 5-6:** Voortgezette ontwikkeling en testen
- **Week 7-8:** Integratie, testen en optimalisatie
- **Week 9-10:** Oplevering en evaluatie

5. Risicobeheer

- **Risico:** Onverwachte persoonlijke omstandigheden
    - **Maatregel:** Regelmatige communicatie en aanpassing van taken indien nodig.
- **Risico:** Technische problemen bij het samenvoegen van de repositories
    - **Maatregel:** Tijdige identificatie en oplossen van conflicten, backup plannen.
- **Risico:** Tijdsdruk door het missen van de eerste deadline
    - **Maatregel:** Strakke planning en prioriteren van belangrijke taken.

6. Communicatie

- Regelmatige teamvergaderingen via Discord.
- Dagelijkse stand-up meetings om de voortgang te bespreken.
- Documentatie van beslissingen en voortgang in een gedeelde online omgeving.

TECHNISCHE REQUIREMENTS

Technische requirements beschrijven de technische specificaties en infrastructuur die nodig zijn om een applicatie effectief te ontwikkelen, implementeren en onderhouden. Het heeft ook architectuur, prestaties, schaalbaarheid en onder houdbaarheid.

ARCHITECTUUR

- **ASP.NET CORE WEB APP:**
  We hebben web app gemaakt met ASP.NET (Model View Controller)
- **API gebaseerd op controllers:**
  De controllers zijn nog niet gebaseerd op Api maar dat gaan we erbij doen wanneer alles

PROGRAMEER TAAL EN FRAMEWORKS

- C#
- ASP.NET Core Mvc
- Entity Framework Core

DATABASE

- SQL Lite
- Taballen voor dieren, categorieën , verblijven en gebruikers
- Entity Framework Core ORM

BEVEILIGING

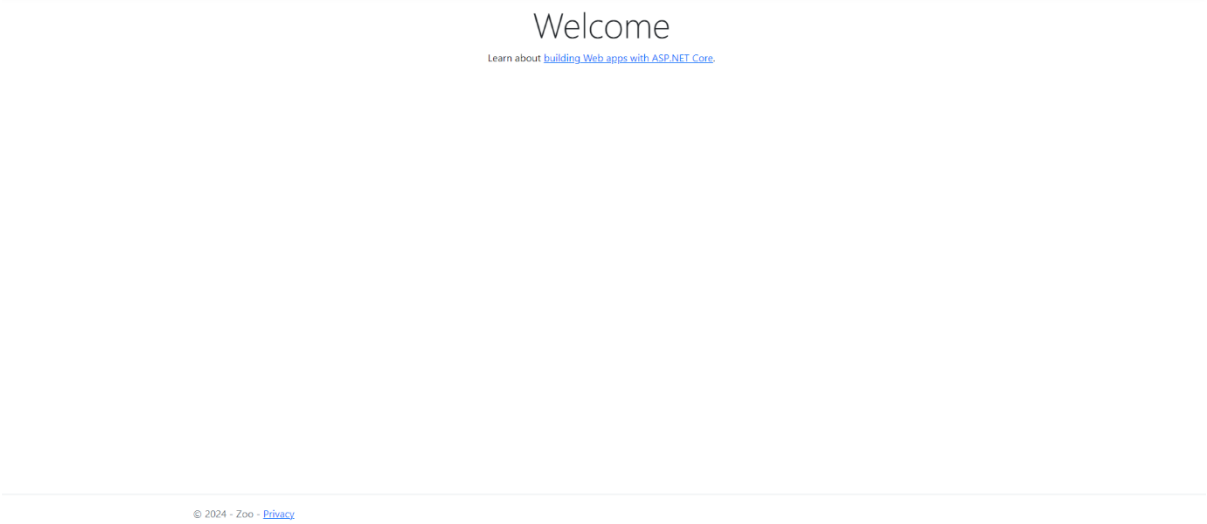
- ASP.NET Core Identity: We hebben ASP.NET Core Identity gebruikt voor gebruikersbeheer en beveiliging.
- Versleutelde gegevensopslag: Alle gevoelige gegevens worden versleuteld opgeslagen.
- GDPR-naleving: De applicatie voldoet aan GDPR-richtlijnen voor gegevensbescherming.

LOGGING EN MONITORING

- Microsoft.Extensions.Logging: We hebben Microsoft.Extensions.Logging gebruikt voor het loggen van applicatiegebeurtenissen.
- Monitoring-tools: Monitoring-tools zijn geïmplementeerd om de gezondheid en prestaties van de applicatie te Uml Diagrammen

Mock ups

Home page,



Op de mockup is een eenvoudige welkomstpagina te zien die is opgebouwd met ASP.NET Core. De pagina bevat de volgende elementen:

1. **Titel "Welcome"**: Groot en gecentreerd in het midden van de pagina.
2. **Link "Learn about building Web apps with ASP.NET Core"**: Onder de titel bevindt zich een hyperlink die naar documentatie of informatie leidt over het bouwen van webapplicaties met ASP.NET Core.
3. **Voettekst**: Onderaan de pagina is er een voettekst die bestaat uit de tekst "© 2024 - Zoo - Privacy". Deze voettekst bevat ook een link naar een privacybeleid.

De pagina heeft een eenvoudige en minimale opmaak, zonder verdere inhoud of opmaakkenmerken zoals afbeeldingen, kleuren, of aanvullende navigatie-elementen.

ANIMAL PAGE,

Animals

Create New

Name	Species	Size	DietaryClass	ActivityPattern	Prey	Enclosure	SpaceRequirement	SecurityRequirement	Category	
Tatyana	ipsam	Medium	Herbivore	Crepuscular	Michelle	60	11.74237527721423	Low	84	Edit | Details | Delete
Coralie	placeat	Small	Herbivore	Nocturnal	Jordan	56	2.076582605008441	Maximum	79	Edit | Details | Delete
Joy	ut	Medium	Insectivore	Crepuscular	Betty	58	18.55992430482791	High	79	Edit | Details | Delete
Maudie	ea	Small	Herbivore	Cathemeral	Dasia	59	16.783213271360374	High	79	Edit | Details | Delete
Kade	atque	Small	Insectivore	Diurnal	Joy	57	4.470711928865915	Low	79	Edit | Details | Delete
Colt	repellat	Small	Herbivore	Diurnal	Clement	60	5.819844440246859	Low	81	Edit | Details | Delete
Mervin	adipisci	Small	Insectivore	Crepuscular	Ole	57	6.312492938139639	Medium	83	Edit | Details | Delete
Kennith	aliquid	Large	Herbivore	Diurnal	Dasia	59	19.543723476180176	Medium	82	Edit | Details | Delete
Jarrod	dolor	ExtraLarge	Herbivore	Diurnal	Coralie	59	6.268831263931066	Low	79	Edit | Details | Delete
Ole	itaque	ExtraLarge	Insectivore	Crepuscular	Rogelio	60	14.002373655489933	High	85	Edit | Details | Delete
Clement	ut	ExtraLarge	Carnivore	Diurnal	Adolfo	57	4.584224425371017	High	85	Edit | Details | Delete
Ulises	dolorum	Small	Omnivore	Cathemeral	Coralie	56	4.865005529164425	Medium	83	Edit | Details | Delete
Rogelio	dignissimos	Small	Herbivore	Nocturnal	Maudie	59	1.5526372625122364	Maximum	82	Edit | Details | Delete
Adolfo	voluptates	Large	Carnivore	Diurnal	Coralie	57	12.335115205850796	Medium	85	Edit | Details | Delete
Maribel	dicta	Large	Herbivore	Nocturnal	Mervin	60	4.015180575173074	Low	84	Edit | Details | Delete
Dasia	vel	Small	Insectivore	Diurnal	Joy	58	6.1785831782842076	Medium	81	Edit | Details | Delete
Jordan	hic	ExtraLarge	Omnivore	Diurnal	Betty	59	13.933525347519753	High	81	Edit | Details | Delete

OP DEZE MOCKUP IS EEN WEBPAGINA TE ZIEN DIE EEN LIJST MET DIEREN WEERGEEFT IN EEN TABEL.

TITEL EN NAVIGATIE:

- **TITEL "ANIMALS"**: GROOT EN BOVENAAN DE PAGINA GECENTREERD.
- **LINK "CREATE NEW"**: EEN HYPERLINK BOVEN DE TABEL WAARMEE EEN NIEUWE DIERENRECORD KAN WORDEN AANGEMAAKT.

Tabel met Diereninformatie:

Het tabel bevat meerdere kolommen met de volgende kopteksten en gegevens voor elk dier:

- **Name**: De naam van het dier.
- **Species**: De soort van het dier.
- **Size**: De grootte van het dier (Small, Medium, Large, ExtraLarge).
- **DietaryClass**: De dieetklasse van het dier (Herbivore, Carnivore, Insectivore, Omnivore).
- **ActivityPattern**: Het activiteitspatroon van het dier (Diurnal, Nocturnal, Crepuscular, Cathemeral).
- **Prey**: De prooi van het dier.
- **Enclosure**: Het nummer van het verblijf waarin het dier zich bevindt.
- **SpaceRequirement**: De ruimtelijke vereisten van het dier, weergegeven als een numerieke waarde.
- **SecurityRequirement**: Het beveiligingsniveau dat nodig is voor het dier (Low, Medium, High, Maximum).
- **Category**: De categorie waartoe het dier behoort, weergegeven met een numerieke waarde.

Acties:

Aan de rechterkant van elke rij in de tabel zijn er drie hyperlinks voor acties:

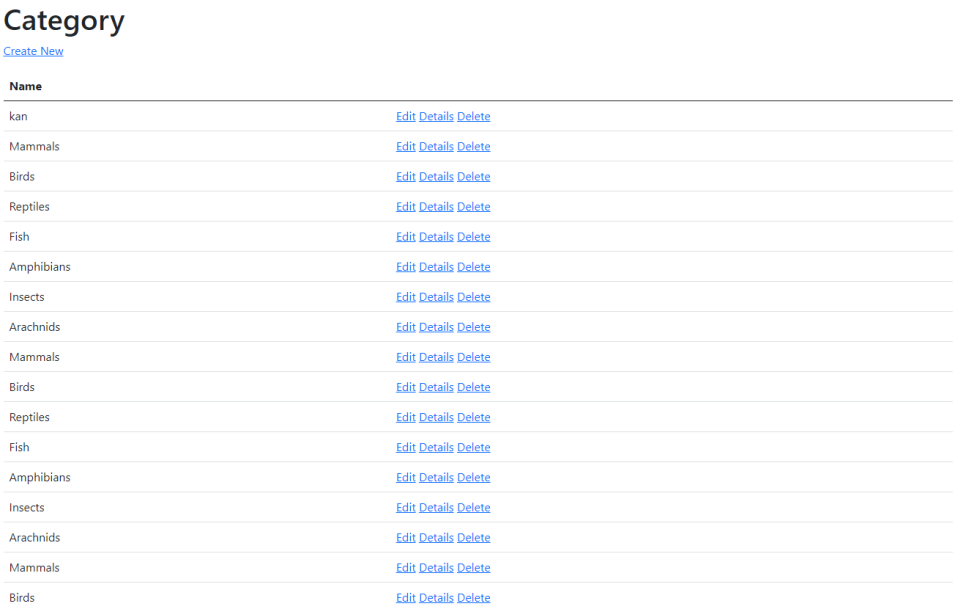
- **Edit**: Een link om de gegevens van het dier te bewerken.
- **Details**: Een link om de details van het dier te bekijken.
- **Delete**: Een link om het dier te verwijderen.

Layout:

- De gegevens worden in een nette en gestructureerde tabel weergegeven.
- De links voor acties zijn per rij duidelijk zichtbaar en bruikbaar.

De pagina is bedoeld om eenvoudig informatie over dieren te bekijken, nieuwe records toe te voegen, en bestaande records te bewerken of te verwijderen.

CATEGORY PAGE,



OP DEZE MOCKUP IS EEN WEBPAGINA TE ZIEN DIE EEN LIJST MET CATEGORIEËN WEERGEEFT IN EEN TABEL. DE DETAILS VAN DE PAGINA:

TITEL EN NAVIGATIE:

- **TITEL "CATEGORY"**: GROOT EN BOVENAAN DE PAGINA GECENTREERD.
- **LINK "CREATE NEW"**: EEN HYPERLINK BOVEN DE TABEL WAARMEE EEN NIEUWE CATEGORIE KAN WORDEN AANGEMAAKT.

TABEL MET CATEGORIEËN:

DE TABEL BEVAT ÉÉN KOLOM MET DE VOLGENDE KOPTEKST EN GEGEVENS VOOR ELKE CATEGORIE:

- **NAME**: DE NAAM VAN DE CATEGORIE. VOORBEELDEN ZIJN "MAMMALS", "BIRDS", "REPTILES", "FISH", "AMPHIBIANS", "INSECTS", "ARACHNIDS", ENZOVOORT.

ACTIES:

AAN DE RECHTERKANT VAN ELKE RIJ IN DE TABEL ZIJN ER DRIE HYPERLINKS VOOR ACTIES:

- **EDIT**: EEN LINK OM DE GEGEVENS VAN DE CATEGORIE TE BEWERKEN.
- **DETAILS**: EEN LINK OM DE DETAILS VAN DE CATEGORIE TE BEKIJKEN.
- **DELETE**: EEN LINK OM DE CATEGORIE TE VERWIJDEREN.

LAYOUT:

- DE GEGEVENS WORDEN IN EEN NETTE EN GESTRUCTUREERDE TABEL WEERGEGEVEN.
- DE LINKS VOOR ACTIES ZIJN PER RIJ DUIDELIJK ZICHTBAAR EN BRUIKBAAR.

DE PAGINA IS BEDOELD OM EENVOUDIG INFORMATIE OVER CATEGORIEËN TE BEKIJKEN, NIEUWE RECORDS TOE TE VOEGEN, EN BESTAANDE RECORDS TE BEWERKEN OF TE VERWIJDEREN.

Enclosure PAGE,

Enclosure

Create New

Name	Climate	HabitatType	SecurityLevel	Size	
ratione	Tropical	Mountain	Low	53.744992925160375	Edit | Details | Delete
dignissimos	Temperate	Aquatic	High	181.50911446753105	Edit | Details | Delete
architecto	Tropical	Mountain	Medium	105.76785946547442	Edit | Details | Delete
a	Continental	Grassland	Medium	112.44869400957838	Edit | Details | Delete
nulla	Temperate	Mountain	Medium	162.77567190902909	Edit | Details | Delete
atque	Polar	Aquatic	Low	161.24986725574988	Edit | Details | Delete
aut	Polar	Grassland	Maximum	164.5902324215236	Edit | Details | Delete
vitae	Tropical	Mountain	Low	130.72713685102258	Edit | Details | Delete
rerum	Temperate	Forest	Low	27.06101302781908	Edit | Details | Delete
rerum	Polar	Desert	Maximum	156.48359942578736	Edit | Details | Delete
optio	Temperate	Mountain	High	101.51652870543627	Edit | Details | Delete
et	Tropical	Forest	Maximum	194.08107347587017	Edit | Details | Delete
molestias	Tropical	Grassland	High	114.46609687052911	Edit | Details | Delete
error	Temperate	Wetland	Low	152.53761013548575	Edit | Details | Delete
adipisci	Temperate	Wetland	Medium	22.65550856693739	Edit | Details | Delete
quia	Tropical	Mountain	Medium	86.28424981467148	Edit | Details | Delete
laboriosam	Continental	Mountain	Maximum	158.27391011726388	Edit | Details | Delete

Op deze mockup is een webpagina te zien die een lijst met verblijven (enclosures) weergeeft in een tabel. De details van de Enclosure pagina:

Titel en Navigatie:

- **Titel "Enclosure"**: Groot en bovenaan de pagina gecentreerd.
- **Link "Create New"**: Een hyperlink boven de tabel waarmee een nieuw verblijf kan worden aangemaakt.

Tabel met Verblijven:

De tabel bevat meerdere kolommen met de volgende kopteksten en gegevens voor elk verblijf:

- **Name**: De naam van het verblijf.
- **Climate**: Het klimaat van het verblijf (bijv. Tropical, Temperate, Polar, etc.).
- **HabitatType**: Het type habitat van het verblijf (bijv. Mountain, Aquatic, Grassland, Forest, Wetland, Desert).
- **SecurityLevel**: Het beveiligingsniveau dat nodig is voor het verblijf (Low, Medium, High, Maximum).
- **Size**: De grootte van het verblijf, weergegeven als een numerieke waarde.

Acties:

Aan de rechterkant van elke rij in de tabel zijn er drie hyperlinks voor acties:

- **Edit**: Een link om de gegevens van het verblijf te bewerken.
- **Details**: Een link om de details van het verblijf te bekijken.
- **Delete**: Een link om het verblijf te verwijderen.

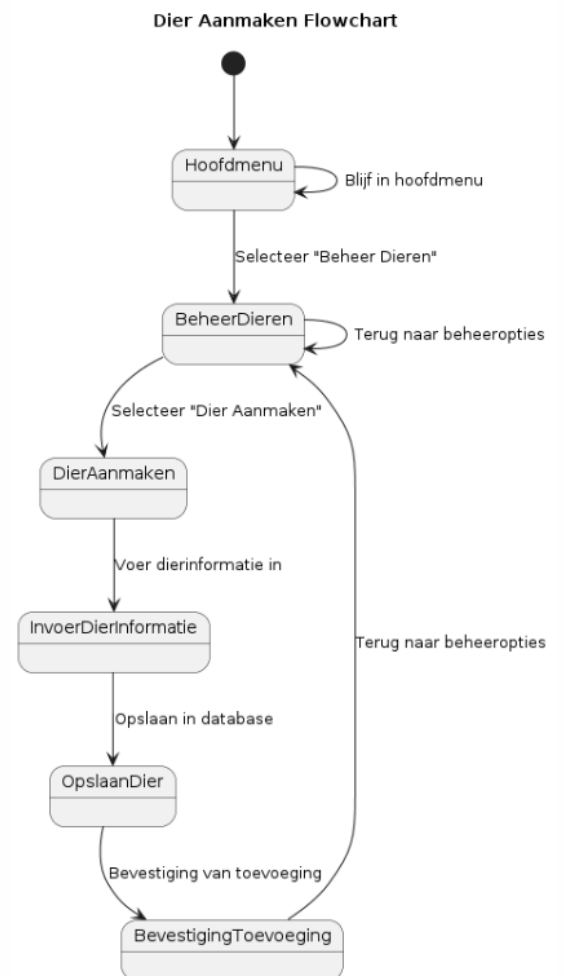
Layout:

- De gegevens worden in een nette en gestructureerde tabel weergegeven.
- De links voor acties zijn per rij duidelijk zichtbaar en bruikbaar.

De pagina is bedoeld om eenvoudig informatie over verblijven te bekijken, nieuwe records toe te voegen, en bestaande records te bewerken of te verwijderen.

FLOWDIAGram

Een flowchart is een grafische weergave van een proces of een systeem, die de stappen van het proces in volgorde weergeeft. Het is een hulpmiddel dat wordt gebruikt om processen, beslissingspunten en de stroom van taken duidelijk en overzichtelijk te illustreren. Flowcharts worden vaak gebruikt in projectmanagement, systeemontwerp, programmering en vele andere gebieden om complexiteit te vereenvoudigen en communicatie te verbeteren.



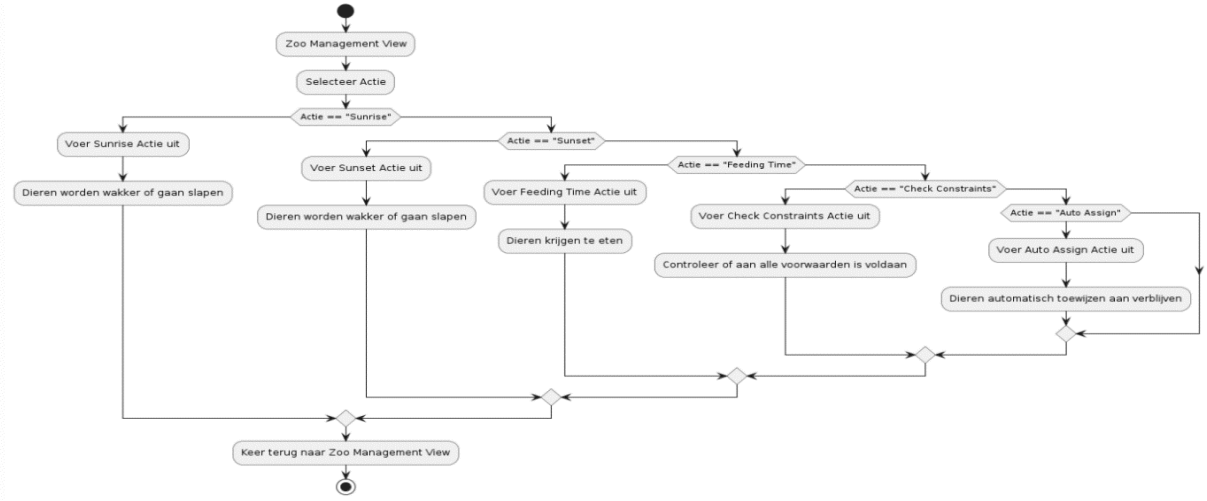
Deze flowchart toont de stappen die een gebruiker doorloopt om een nieuw dier toe te voegen aan het systeem, inclusief het navigeren door de menu's, het invoeren en opslaan van dierinformatie, en het ontvangen van een bevestiging van de succesvolle toevoeging.

ACTIVITY DIAGRAM

WAT IS EEN ACTIVITY DIAGRAM?

In een activity diagram worden activiteiten voorgesteld door rechthoeken, terwijl de overgangen tussen activiteiten worden weergegeven door pijlen die de richting van de stroom aangeven. De activiteiten kunnen verschillende acties vertegenwoordigen, zoals het uitvoeren van berekeningen, het nemen van beslissingen, het wachten op input, enzovoort.

Het begint meestal met een cirkel bovenaan, dat is waar de activiteit begint, en eindigt met een cirkel onderaan, dat is waar de activiteit eindigt. Wordt meestal gebruikt om de stroom van activiteiten of processen in een systeem te visualiseren.



Startpunt: Het proces begint bij het startpunt.

Zoo Management View: De gebruiker bevindt zich op de hoofdpagina voor het beheren van de dierentuin.

Selecteer Actie: De gebruiker kan kiezen uit verschillende acties:

Sunset: Deze actie bepaalt welke dieren wakker worden en welke gaan slapen bij zonsondergang.

Feeding Time: Deze actie zorgt ervoor dat dieren te eten krijgen.

Check Constraints: Deze actie controleert of aan alle voorwaarden voor de dieren en hun verblijven is voldaan.

Uitvoeren van de Actie: Afhankelijk van de geselecteerde actie wordt de specifieke taak uitgevoerd.

Keer terug naar Zoo Management View: Na het uitvoeren van de actie keert de gebruiker terug naar de hoofdpagina.

Stoppunt: Het proces eindigt hier.

CLASS DIAGRAM/ERD

WAT IS EEN CLASS DIAGRAM?

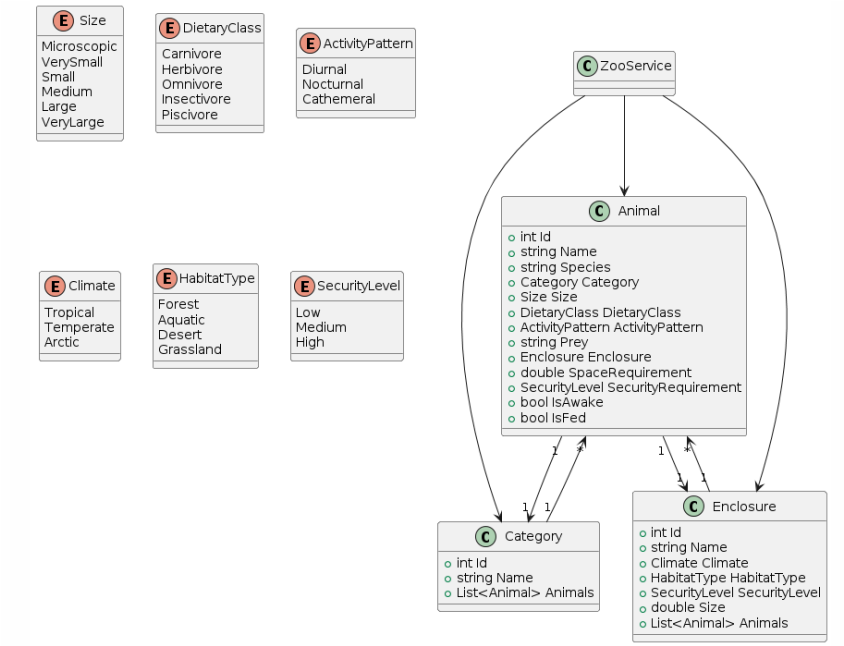
Een klassendiagram is een type diagram in de Unified Modeling Language (UML) dat wordt gebruikt om de statische structuur van een systeem te visualiseren. Het richt zich op de klassen in een systeem, hun attributen (eigenschappen) en methoden (gedrag), en de relaties tussen deze klassen.

KLASSEN: Een klasse vertegenwoordigt een concept, entiteit of object in het systeem dat we modelleren. Het bevat attributen die de eigenschappen van het object beschrijven, en methoden die het gedrag of de operaties van het object definiëren.

ATTRIBUTEN: Attributen zijn eigenschappen van een klasse. Ze vertegenwoordigen gegevens die door objecten van die klasse worden bijgehouden. Attributen worden meestal weergegeven onder de naam van de klasse in een klassendiagram.

METHODEN: Methoden zijn functies of operaties die door objecten van een klasse kunnen worden uitgevoerd. Ze beschrijven het gedrag van objecten en vertegenwoordigen de acties die een object kan uitvoeren. Methoden worden vaak weergegeven onder de attributen van een klasse in een klassendiagram.

RELATIES: Relaties definiëren de verbanden tussen klassen. Er zijn verschillende soorten relaties, waaronder associatie, aggregatie, compositie, generalisatie (of overerving) en afhankelijkheid. Relaties worden weergegeven met lijnen die klassen verbinden en worden vaak voorzien van labels om de aard van de relatie aan te geven



ANIMAL

Attributen:

- int Id: Unieke identificatie van het dier.
- string Name: Naam van het dier.
- string Species: Soort van het dier.
- Category Category: Categorie waartoe het dier behoort.
- Size Size: Grootte van het dier.
- DietaryClass DietaryClass: Voedingsklasse van het dier.
- ActivityPattern ActivityPattern: Activiteitspatroon van het dier.
- string Prey: Prooi van het dier .
- Enclosure Enclosure: Verblijf van het dier.
- double SpaceRequirement: Ruimtevereiste voor het dier.
- SecurityLevel SecurityRequirement: Beveiligingsniveau voor het dier.
- bool IsAwake: Of het dier wakker is.
- bool IsFed: Of het dier gevoed is.

CATEGORY

Attributen:

- int Id: Unieke identificatie van de categorie.
- string Name: Naam van de categorie.
- List<Animal> Animals: Lijst van dieren die tot deze categorie behoren.

ENCLOSURE

Attributen:

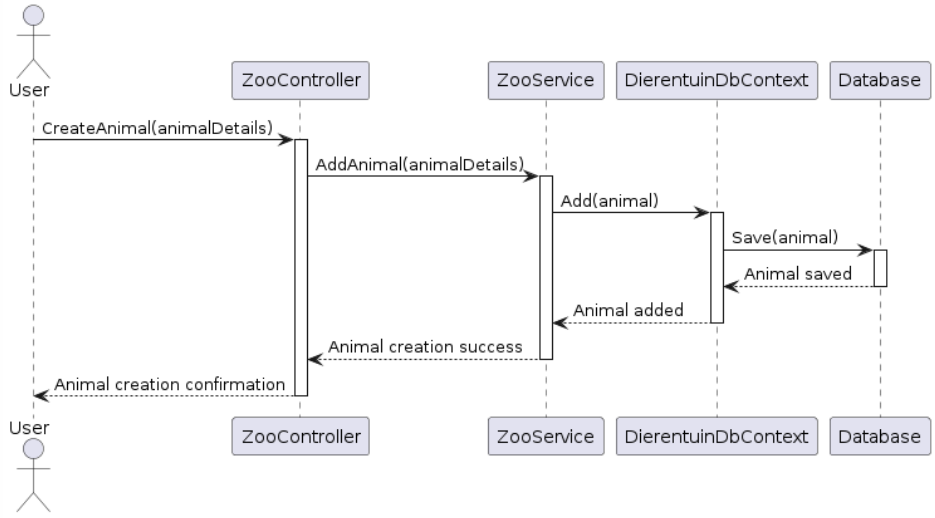
- int Id: Unieke identificatie van het verblijf.
- string Name: Naam van het verblijf.
- Climate Climate: Klimaat van het verblijf.
- HabitatType HabitatType: Type habitat van het verblijf.
- SecurityLevel SecurityLevel: Beveiligingsniveau van het verblijf.
- double Size: Grootte van het verblijf.
- List<Animal> Animals: Lijst van dieren in dit verblijf.

Door het diagram kunnen ze ook zien welke dieren bij elkaar horen in dezelfde categorie en welke speciale eisen elk dier heeft, zoals hoeveel ruimte ze nodig hebben en welk beveiligingsniveau ze vereisen. Bovendien helpt het hen te begrijpen in welk klimaat en type habitat elk dier het beste gedijt. Dit alles maakt het beheer van de dierentuin efficiënter en zorgt ervoor dat de dieren goed verzorgd worden en in een geschikte omgeving leven.

SEQUENCE DIAGRAM

WAT IS EEN SEQUENCE DIAGRAM?

Een sequence diagram is een soort diagram dat wordt gebruikt om te laten zien hoe verschillende onderdelen van een systeem met elkaar communiceren en samenwerken in een bepaalde volgorde. Het is alsof je een stripverhaal maakt van hoe verschillende personages met elkaar praten en acties ondernemen. Sequence diagrammen zijn handig omdat ze laten zien hoe een systeem werkt in termen van berichten die heen en weer worden gestuurd tussen verschillende onderdelen. Ze worden vaak gebruikt in softwareontwikkeling om het gedrag van een systeem te begrijpen, problemen op te lossen en te communiceren met stakeholders.



1. **Gebruiker initieert:** De gebruiker start het proces door een verzoek te sturen naar de ZooController om een nieuw dier aan te maken met de gegeven details.
2. **Controller roept service aan:** De ZooController roept de AddAnimal methode aan op de ZooService met de details van het nieuwe dier.
3. **Service voegt dier toe aan database context:** De ZooService voegt het dier toe aan de database context (DierentuinDbContext).
4. **Database slaat dier op:** De DierentuinDbContext slaat het dier op in de database.
5. **Bevestiging van opslag**: De database bevestigt dat het dier succesvol is opgeslagen.
6. **Service ontvangt bevestiging:** De DierentuinDbContext stuurt een bevestiging terug naar de ZooService dat het dier is toegevoegd.
7. **Controller ontvangt bevestiging:** De ZooService stuurt een succesmelding terug naar de ZooController.
8. **Gebruiker ontvangt bevestiging**: De ZooController stuurt een bevestiging van de succesvolle aanmaak van het dier terug naar de gebruiker.

COMPONENTENDETAIL

WAT IS HET?

Een componentendetail is eigenlijk een gedetailleerde beschrijving van de verschillende stukjes software die samen een programma vormen. Het vertelt je wat elk stukje doet, hoe het werkt, en hoe het met de andere stukjes samenwerkt.

DIEREN COMPONENT

WAT DOET HET?

- Beheren van dieren: toevoegen, bekijken, aanpassen en verwijderen.
- Zoeken en filteren van dieren op basis van eigenschappen zoals soort en naam.
- Koppelen van dieren aan categorieën en verblijven.

HOE WERKT HET?

WEBPAGINA'S:

- **Index pagina:** toont een lijst met alle dieren.
- **Create pagina:** voor het toevoegen van nieuwe dieren.
- **Edit pagina:** voor het bewerken van bestaande dieren.
- **Details pagina:** voor het bekijken van details van een dier.

GEBRUIKTE TECHNOLOGIE:

- Programmeertaal: C#
- Framework: ASP.NET Core MVC
- Tools: Entity Framework Core voor database-interacties.

CATEGORIEËN COMPONENT

WAT DOET HET?

- Beheren van diercategorieën: toevoegen, bekijken, aanpassen en verwijderen.
- Toewijzen van dieren aan categorieën.

HOE WERKT HET?

WEBPAGINA'S:

- Index pagina: toont een lijst met alle categorieën.
- Create pagina: voor het toevoegen van nieuwe categorieën.
- Edit pagina: voor het bewerken van bestaande categorieën.
- Details pagina: voor het bekijken van details van een categorie.

Verblijven Component

Wat doet het?

1. Beheren van verblijven: toevoegen, bekijken, aanpassen en verwijderen.
2. Toewijzen van dieren aan verblijven.
3. Beheren van verblijfsinformatie zoals naam, klimaat, habitat en grootte.

Hoe werkt het?

Webpagina's:

- **Index pagina**: toont een lijst met alle verblijven.
- **Create pagina**: voor het toevoegen van nieuwe verblijven.
- **Edit pagina:** voor het bewerken van bestaande verblijven.
- **Details pagina:** voor het bekijken van details van een verblijf.

Database Component

Wat doet het?

- Beheert de database structuur en zorgt voor het uitvoeren van wijzigingen (migraties).
- Initieert de database met startgegevens (seed data).
- Hoe werkt het?

Afhankelijk van:

1. Programmeertaal: C#
2. Framework: Entity Framework Core

TEST/UNITTESTING

Unittesting, ook wel bekend als unit testing, is een manier om kleine stukjes code, genaamd "units," afzonderlijk te testen om ervoor te zorgen dat ze correct werken.

In softwareontwikkeling verwijst een "unit" meestal naar een enkele functie, methode of klasse in een programma.

Het hoofddoel van unittesting is om fouten vroegtijdig op te sporen en te corrigeren. Door afzonderlijke onderdelen van de code te testen, kunnen ontwikkelaars problemen identificeren voordat de code wordt geïntegreerd met andere delen van het programma.

TEST 1

De Index_ReturnsViewResult_WithAListOfDieren test valideert de functionaliteit van de Index methode in de DierenController:

```
[Fact]
0 references
public void Index_ReturnsViewResult_WithAListOfDieren()
{
    // Act
    var result = _controller.Index();

    // Assert
    var viewResult = Assert.IsType<ViewResult>(result);
    var model = Assert.IsAssignableFrom<IEnumerable<Dieren>>(viewResult.ViewData.Model);
    Assert.Equal(2, model.Count());
}
```

Deze test is belangrijk om te controleren of de Index methode goed werkt. Het doet dit door:

- Te zorgen dat de Index methode een view (scherm) teruggeeft met een lijst van dieren.
- Zeker te weten dat het aantal dieren in deze lijst klopt met wat we verwachten.

Als deze test slaagt, weten we dat de Index methode de juiste gegevens teruggeeft en goed samenwerkt met het scherm in de applicatie.

TEST 2

De test Details_ReturnsViewResult_WithDieren heeft als doel te controleren of de Details methode in de DierenController correct functioneert.

```
[Fact]
0 references
public void Details_ReturnsViewResult_WithDieren()
{
    // Act
    var result = _controller.Details(1);

    // Assert
    var viewResult = Assert.IsType<ViewResult>(result);
    var model = Assert.IsAssignableFrom<Dieren>(viewResult.ViewData.Model);
    Assert.Equal("Lion", model.Name);
}
```

BELANG VAN DE TEST

Deze test is essentieel om te garanderen dat de Details methode correct functioneert door:

- Te zorgen dat de Details methode een view (scherm) teruggeeft.
- Zeker te weten dat het teruggegeven dier de juiste gegevens bevat, zoals de naam "Lion" voor het dier met ID 1.

Door deze test te laten slagen, kan worden verzekerd dat de Details methode de juiste gegevens retourneert en correct samenwerkt met de view binnen de applicatie.

TEST 3

De test Details_ReturnsNotFound_WhenIdIsInvalid controleert of de Details methode in de DierenController correct omgaat met situaties waarin een ongeldig ID wordt opgegeven

BELANG VAN DE TEST

Deze test is belangrijk om ervoor te zorgen dat de Details methode goed werkt door:

- Correct Afhandelen van Foutieve Invoer: Te controleren dat de Details methode een NotFoundResult teruggeeft als een ongeldig ID wordt opgegeven. Dit laat zien dat de methode goed omgaat met situaties waarin een dier niet bestaat.
- Verbeteren van de Gebruikerservaring: Te zorgen dat de gebruiker een duidelijke foutmelding krijgt als het gevraagde dier niet bestaat.

```
[Fact]
0 references
public void Details_ReturnsNotFound_WhenIdIsInvalid()
{
    // Act
    var result = _controller.Details(3);

    // Assert
    Assert.IsType<NotFoundResult>(result);
}
```

TEAM 4

De test Create_ReturnsViewResult controleert of de Create methode in de DierenController een ViewResult teruggeeft.

```
[Fact]
0 references
public void Create_ReturnsViewResult()
{
    // Act
    var result = _controller.Create();

    // Assert
    Assert.IsType<ViewResult>(result);
}
```

BELANG VAN DE TEST

- Te zorgen dat de Create methode een ViewResult teruggeeft. Dit is belangrijk omdat het bevestigt dat de methode de juiste view retourneert voor het creëren van een nieuw dier.

- Door deze test te laten slagen, kan worden verzekerd dat de Create methode de juiste view weergeeft waarin gebruikers de informatie voor een nieuw dier kunnen invoeren.

TEST 5

De test Create_RedirectsToIndex_OnValidModel controleert of de Create methode in de Dieren Controller een correct RedirectToActionResult teruggeeft wanneer een geldig model wordt aangemaakt.

BELANG VAN DE TEST

Deze test is essentieel om te garanderen dat de Create methode correct functioneert door:

- Verifiëren van Redirect: Te zorgen dat de Create methode een RedirectToActionResult teruggeeft en de gebruiker doorverwijst naar de Index actie na het succesvol aanmaken van een nieuw dier. Dit is belangrijk omdat het aangeeft dat de methode correct omgaat met de invoer en de juiste vervolgactie onderneemt.
- Controleren van Gegevensopslaan: Te verifiëren dat het nieuwe dier daadwerkelijk wordt toegevoegd aan de database en dat de wijzigingen worden opgeslagen. Dit gebeurt door te controleren dat de Add methode van de context één keer wordt aangeroepen en dat SaveChanges één keer wordt aangeroepen. Dit garandeert dat de gegevens persistent zijn.

```
[Fact]
0 references
public async Task Create_RedirectsToIndex_OnValidModel()
{
    // Arrange
    var dier = new Dieren { Id = 3, Name = "Elephant", Species = "Loxodonta" };

    // Act
    var result = _controller.Create(dier);

    // Assert
    var redirectToActionResult = Assert.IsType<RedirectToActionResult>(result);
    Assert.Equal("Index", redirectToActionResult.ActionName);
    _mockSet.Verify(m => m.Add(It.IsAny<Dieren>()), Times.Once);
    _mockContext.Verify(m => m.SaveChanges(), Times.Once);
}
```

Installatie

Vereisten

- Visual Studio 2022 (of nieuwer)
- .NET 7.0 SDK (of nieuwer)
- SQLite database
- NuGet voor het beheren van pakketten

Stap 1: Visual Studio Installeren

- Download en installeer Visual Studio 2022.
- Tijdens de installatie, zorg ervoor dat je de workload "ASP.NET and web development" selecteert.

Stap 2: Het Project Kloon van GitHub

Open Visual Studio 2022.

Ga naar Git > Clone Repository.

Voer de URL van de GitHub repository in https://github.com/ebram101/dierentuinn-devops

Klik op Clone.

Stap 3: Project Configureren

Open het gekloonde project in Visual Studio.

Controleer of de benodigde NuGet-pakketten zijn geïnstalleerd:

- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.Sqlite
- Microsoft.AspNetCore.Mvc

Indien nodig, installeer deze pakketten via Tools > NuGet Package Manager > Manage NuGet Packages for Solution.

Stap 4: Database Instellen

Open het bestand appsettings.json.

Zorg ervoor dat de connection string correct is ingesteld voor SQLite:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=Dierentuin.db"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

Stap 5: Database Migraties Toepassen

Open de Package Manager Console via Tools > NuGet Package Manager > Package Manager Console.

1. Voer de volgende commando's uit om de database te migreren en te vullen met seed data:
2. Add-Migration InitialCreate
3. Update-Database

Stap 6: Applicatie Starten

Druk op F5 of klik op de Start knop in Visual Studio om de applicatie te starten.

De webbrowser wordt geopend en toont de startpagina van het dierentuinbeheersysteem.

Stap 7: Controleer de Seed Data

Open de SQLite database (Dierentuin.db) met een tool zoals DB Browser for SQLite.

Verifieer dat de tabellen correct zijn aangemaakt en gevuld met de seed data.

REFLECTIE

Tijdens dit project hebben we veel geleerd over softwareontwikkeling. We hebben onze kennis van C# en ASP.NET Core MVC verbeterd en geleerd hoe we een webapplicatie goed kunnen opbouwen met controllers, modellen en views. Met Entity Framework Core hebben we geleerd om databases te beheren, inclusief het maken van migraties en het vullen van de database met gegevens. We hebben SQLite gebruikt voor praktische ervaring met een eenvoudige maar krachtige database. Het was ons niet even gelukt om de database vooruit te krijgen. We hebben als groep het proberen op te lossen maar het is ons niet gelukt.

We hebben samengewerkt door middel van branches in Git door iedereen een taak te geven en het verdelen.

Dit project heeft ons een goed begrip gegeven van softwareontwikkeling, van plannen en ontwerpen tot coderen en testen. We hebben waardevolle praktische ervaring opgedaan met essentiële tools en technieken. Het was een leerzame ervaring die onze technische en samenwerkingsvaardigheden heeft versterkt.