Documentatie

Hieu Phan, Abdelaziz Amine Groep 13

Inleiding 3
Doel van de API 3
Hoe is de API gemaakt 4
1. Opzetten van het ASP.NET Core project 4
2. Modellen definiëren 4
3. Database Context 4
4. Controllers schrijven 4
Overzicht van de API 4
1. Ophalen van alle dieren 5
2. Ophalen van een specifiek dier 5
3. Een nieuw dier toevoegen 5
4. Een bestaand dier aanpassen 6
5. Een dier verwijderen 7
6. Acties bij zonsopgang 8
7. Acties bij zonsondergang 8
8. Acties bij voedertijd 9
Conclusie 9
Wireframes 10
ERD 15

Inleiding

Deze documentatie beschrijft de werking van een op ASP.NET Core gebaseerde RESTful API voor het beheren van dieren in een dierentuin. We bespreken de functionaliteit van de API, hoe deze is ontworpen en geïmplementeerd, en hoe je ermee kunt werken.

Doel van de API

De API stelt gebruikers (bijvoorbeeld beheerders van de dierentuin of ontwikkelaars van een front-end interface) in staat om alle interacties met dieren in de dierentuin digitaal te beheren. Dit omvat:

- Het ophalen van een lijst van alle dieren.
- Het ophalen van informatie over een specifiek dier.
- Het toevoegen, wijzigen en verwijderen van dieren.
- Speciale acties, zoals het weergeven van wat dieren doen bij zonsopgang, zonsondergang en voedertijd.

De API volgt de REST-principes en gebruikt standaard HTTP-methodes zoals `GET`, `POST`, `PUT` en `DELETE`.

Hoe is de API gemaakt

Deze API is gebouwd met **ASP.NET Core** en maakt gebruik van de **Entity Framework Core** (EF Core) ORM om te communiceren met de database. Dit waren de stappen die we hebben genomen om de API te ontwikkelen:

1. Opzetten van het ASP.NET Core project

We begonnen met een nieuw ASP.NET Core project in Visual Studio. We hebben gekozen voor een Web API-sjabloon, die standaard een basisstructuur biedt om API's te bouwen.

2. Modellen definiëren

We hebben de modellen gedefinieerd die de gegevensstructuur van de API vertegenwoordigen. In dit geval bevatte ons project modellen zoals `Animal`, `Category`, en `Enclosure`. Elk model komt overeen met een tabel in de database.

3. Database Context

We hebben een **Database Context** (DIERENTUIN13Context) opgezet om toegang tot de database te beheren. Deze context gebruikt EF Core om queries naar de database te sturen en de gegevensstructuur te synchroniseren.

4. Controllers schrijven

De kern van de API zijn de **Controllers**, die de inkomende HTTP-verzoeken verwerken. Voor deze API hebben we een controller genaamd `AnimalsApiController` gemaakt. Deze controller bevat alle endpoints en methodes die je hieronder kunt vinden.

Overzicht van de API

Hier is een overzicht van de beschikbare functionaliteiten van de API. Elk van deze functies kan worden uitgevoerd via specifieke endpoints.

1. Ophalen van alle dieren

Je kunt een lijst ophalen van alle dieren die momenteel in de database staan. Deze lijst bevat ook aanvullende informatie, zoals de categorie en het verblijf van elk dier.

```
[HttpGet]
0 references
public async Task<ActionResult<IEnumerable<Animal>>> GetAnimals()
{
    return await _context.Animal.Include(a => a.Category).Include(a => a.Enclosure).ToListAsync();
}
```

2. Ophalen van een specifiek dier

Met dit endpoint kun je informatie opvragen over een specifiek dier door het ID van dat dier op te geven. Als het dier niet wordt gevonden, krijg je een foutmelding.

```
[HttpGet("{id}")]
0 references
public async Task<ActionResult<Animal>> GetAnimal(int id)
{
    var animal = await _context.Animal.Include(a => a.Category).Include(a => a.Enclosure).FirstOrDefaultAsync(a => a.Id == id);

    if (animal == null)
    {
        return NotFound();
    }

    return animal;
}
```

3. Een nieuw dier toevoegen

Dit stelt je in staat om een nieuw dier toe te voegen aan de database. Hiervoor dien je de benodigde informatie, zoals de naam van het dier en de bijbehorende categorie en verblijf, aan te leveren.

```
[HttpPost]
0 references
public async Task<ActionResult<Animal>> PostAnimal(Animal animal)
{
    _context.Animal.Add(animal);
    await _context.SaveChangesAsync();

    return CreatedAtAction("GetAnimal", new { id = animal.Id }, animal);
}
```

4. Een bestaand dier aanpassen

Met deze functionaliteit kun je de gegevens van een bestaand dier wijzigen. Dit is handig als een dier bijvoorbeeld naar een ander verblijf wordt verplaatst of een nieuwe naam krijgt.

```
[HttpPut("{id}")]
0 references
public async Task<IActionResult> PutAnimal(int id, Animal animal)
{
    if (id != animal.Id)
    {
        return BadRequest();
    }

    _context.Entry(animal).State = EntityState.Modified;

    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!AnimalExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }

    return NoContent();
}
```

5. Een dier verwijderen

Je kunt een dier permanent uit de database verwijderen door het ID van dat dier op te geven. Als het dier niet wordt gevonden, ontvang je een foutmelding.

```
[HttpDelete("{id}")]
0 references
public async Task<IActionResult> DeleteAnimal(int id)
{
    var animal = await _context.Animal.FindAsync(id);
    if (animal == null)
    {
        return NotFound();
    }

    _context.Animal.Remove(animal);
    await _context.SaveChangesAsync();

    return NoContent();
}
```

6. Acties bij zonsopgang

Dit endpoint geeft een lijst van acties die de dieren uitvoeren bij zonsopgang. Elk dier heeft een specifieke actie die wordt uitgevoerd.

```
[HttpGet("Sunrise")]
0 references
public ActionResult<IEnumerable<string>> GetSunriseActions()
{
    var actions = _context.Animal.Select(a => new { a.Name, Action = a.ActieSunrise() }).ToList();
    return Ok(actions);
}
```

7. Acties bij zonsondergang

Hiermee kun je zien welke acties de dieren uitvoeren wanneer de zon ondergaat. Net als bij zonsopgang is deze lijst afhankelijk van de eigenschappen van elk dier.

```
[HttpGet("Sunset")]
0 references
public ActionResult<IEnumerable<string>> GetSunsetActions()
{
    var actions = _context.Animal.Select(a => new { a.Name, Action = a.ActieSunset() }).ToList();
    return Ok(actions);
}
```

8. Acties bij voedertijd

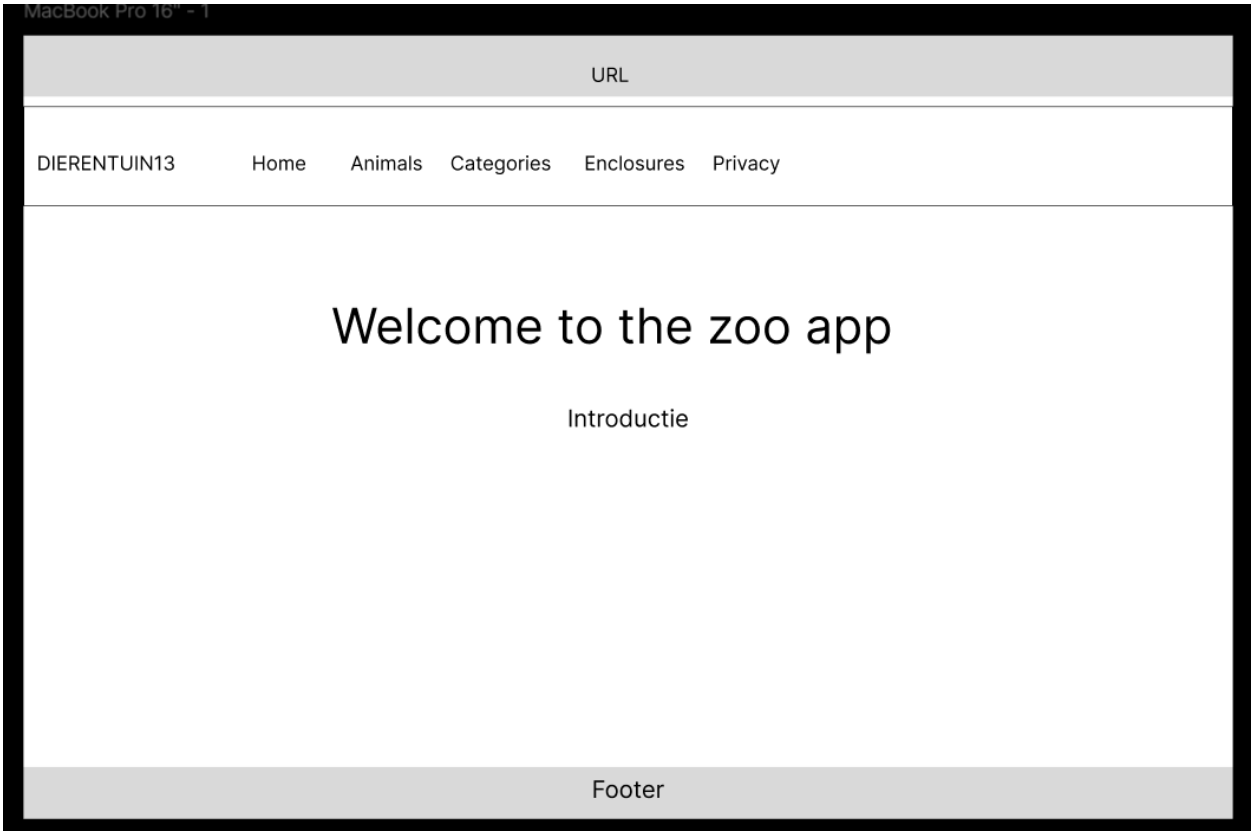
Dit endpoint geeft een overzicht van de activiteiten van dieren tijdens voedertijd. Het laat bijvoorbeeld zien hoe ze reageren op het krijgen van eten.

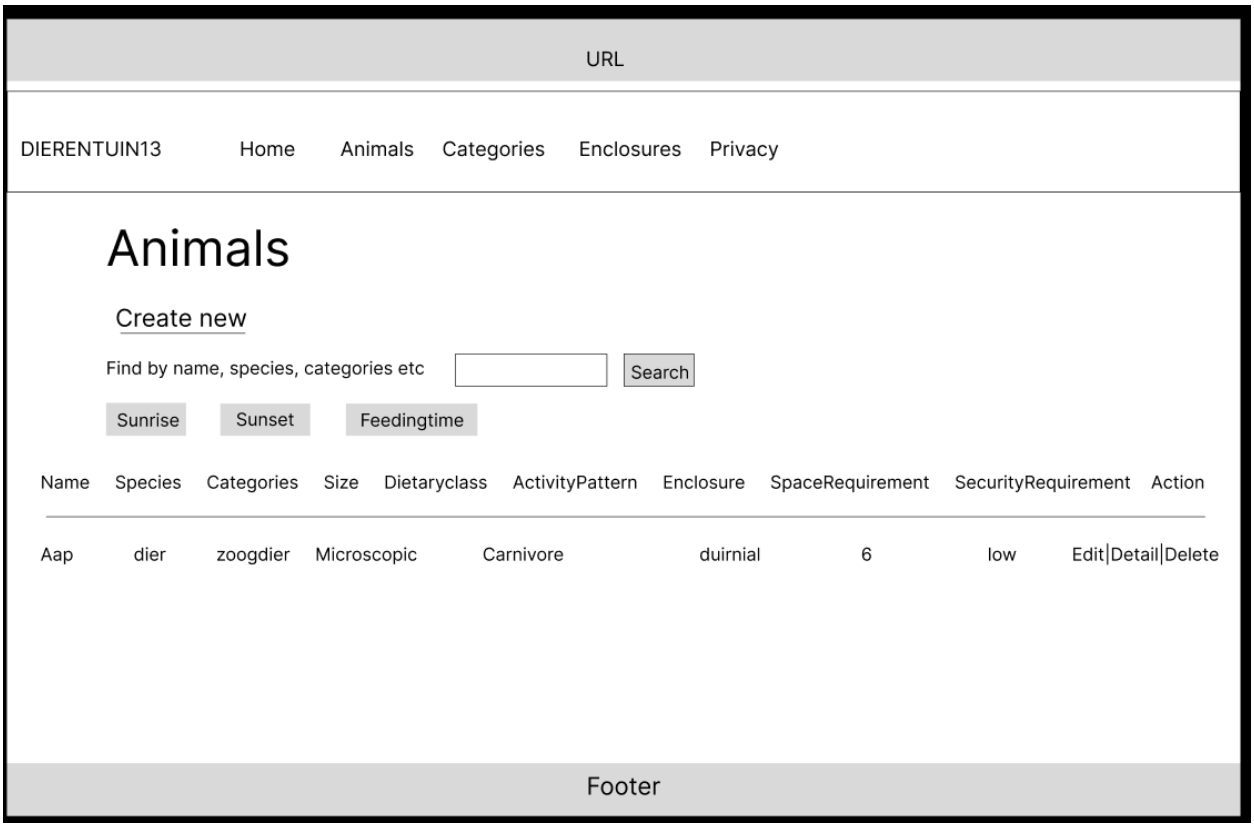
```
[HttpGet("FeedingTime")]
0 references
public ActionResult<IEnumerable<string>> GetFeedingTimeActions()
{
    var actions = _context.Animal.Select(a => new { a.Name, Action = a.ActieFeedingTime() }).ToList();
    return Ok(actions);
}
```

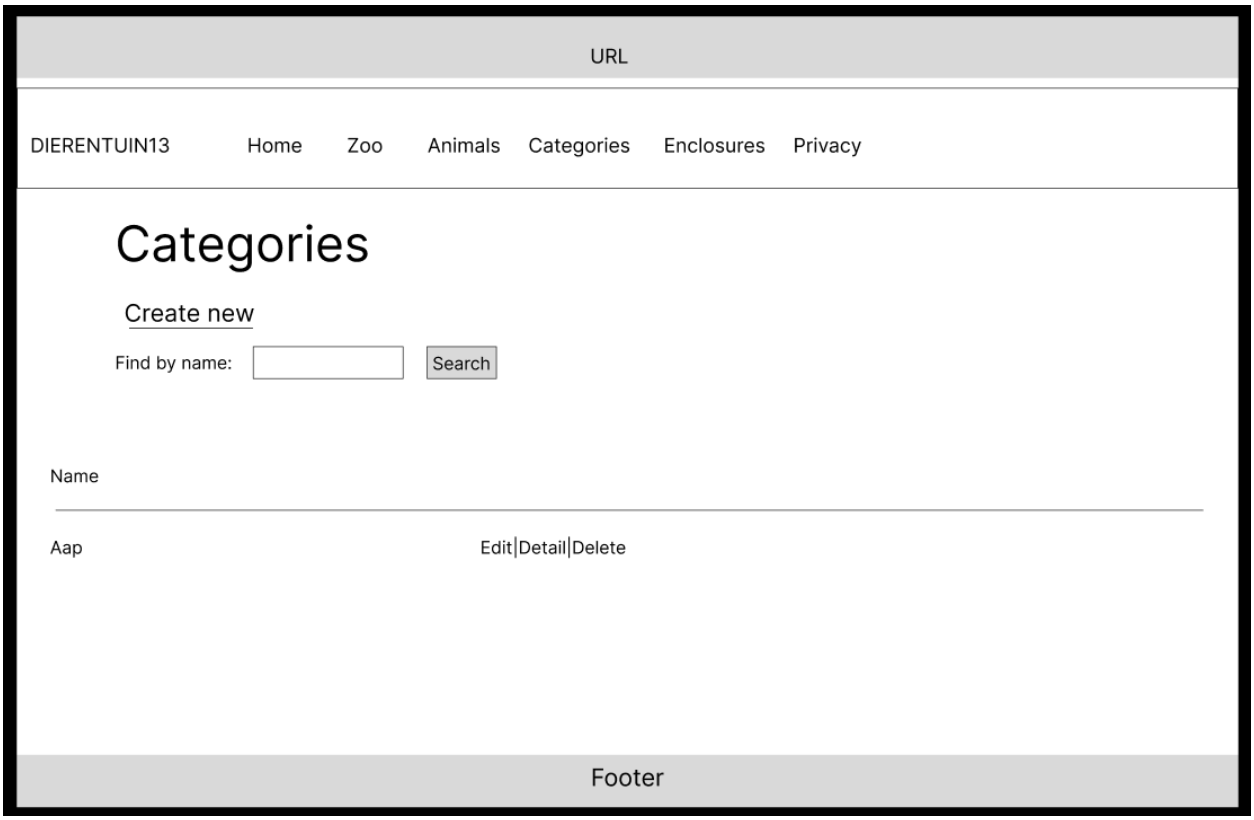
Conclusie

De Dierentuin API biedt een complete set aan tools om dieren te beheren en biedt flexibiliteit om te integreren met andere systemen. De API is ontworpen met het oog op eenvoud, herbruikbaarheid en uitbreidbaarheid. Of je nu een beheerder bent die gegevens wil bijwerken of een ontwikkelaar die werkt aan een nieuwe interface, deze API biedt alle functionaliteit die je nodig hebt.

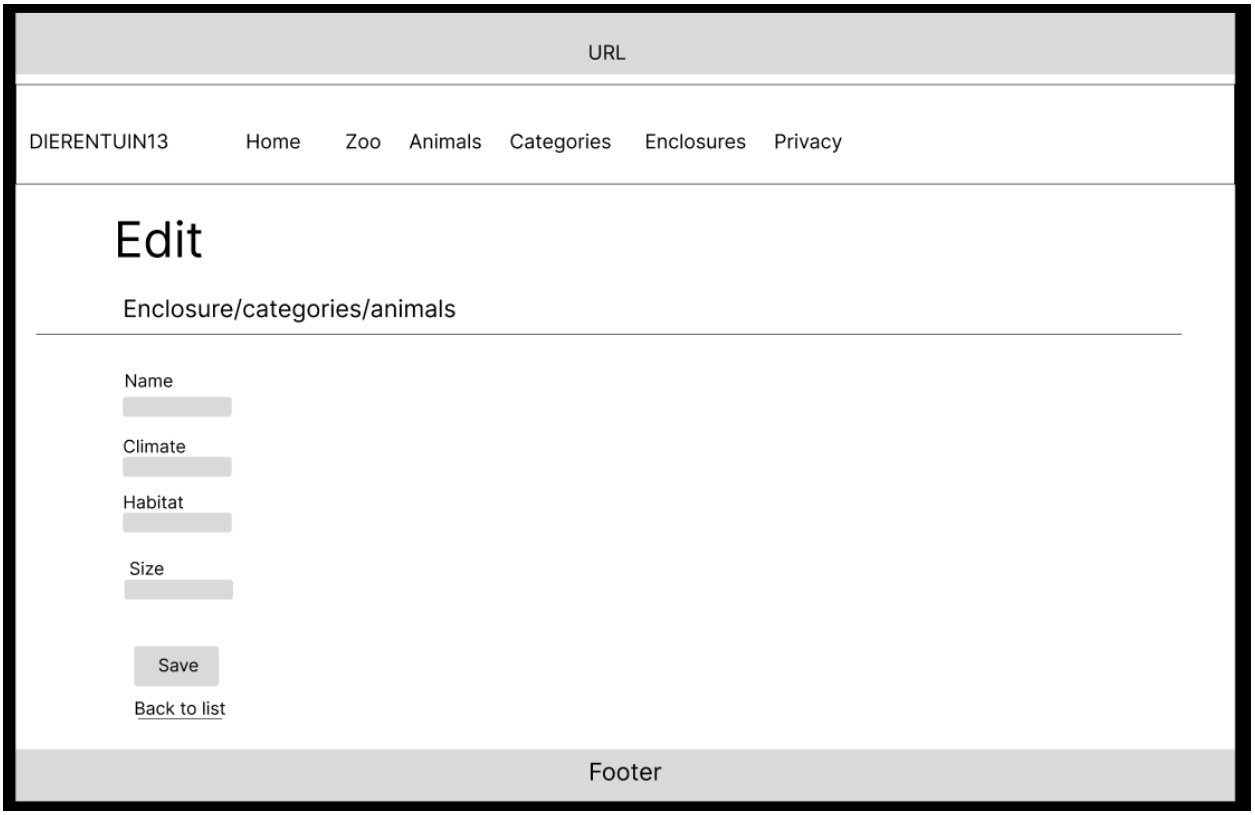
Wireframes

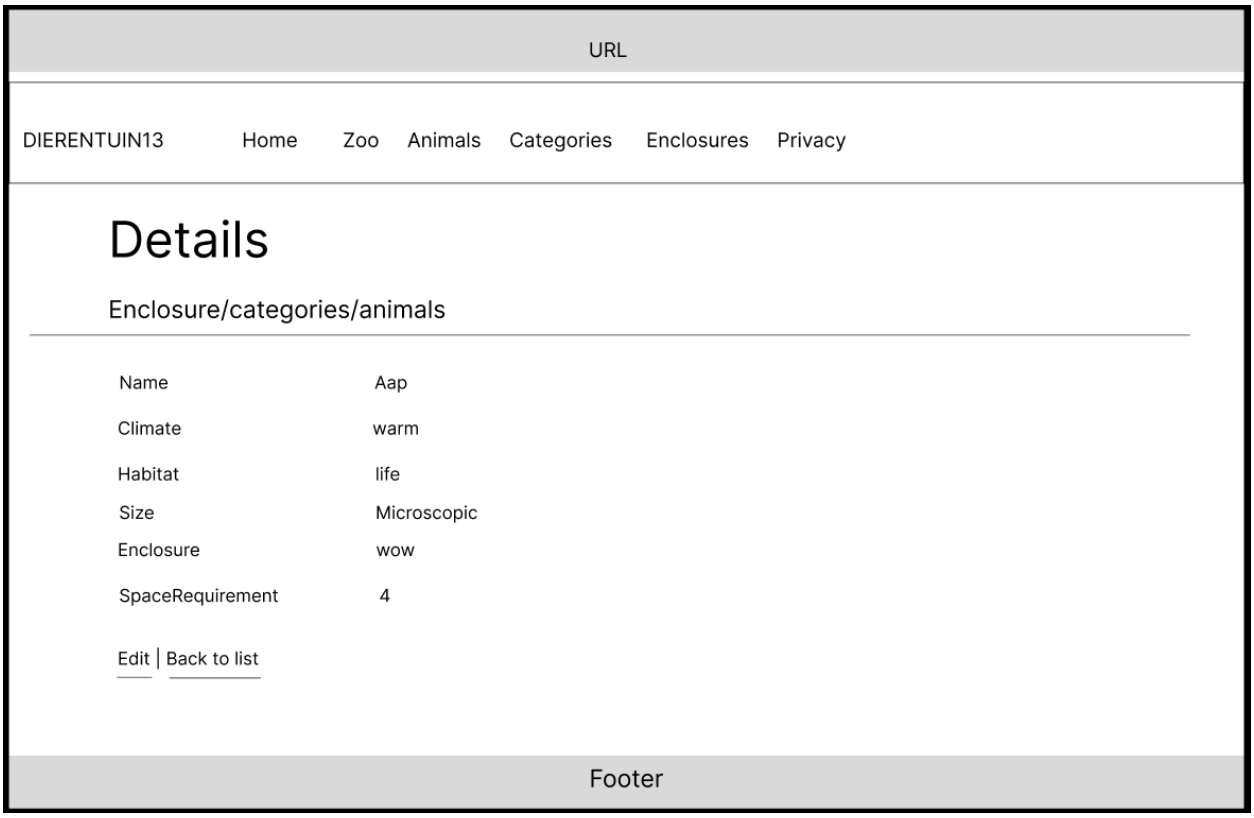

MacBook Pro 16" - 1
URL
DIERENTUIN13
Home
Animals
Categories
Enclosures
Privacy
Welcome to the zoo app
Introductie
Footer

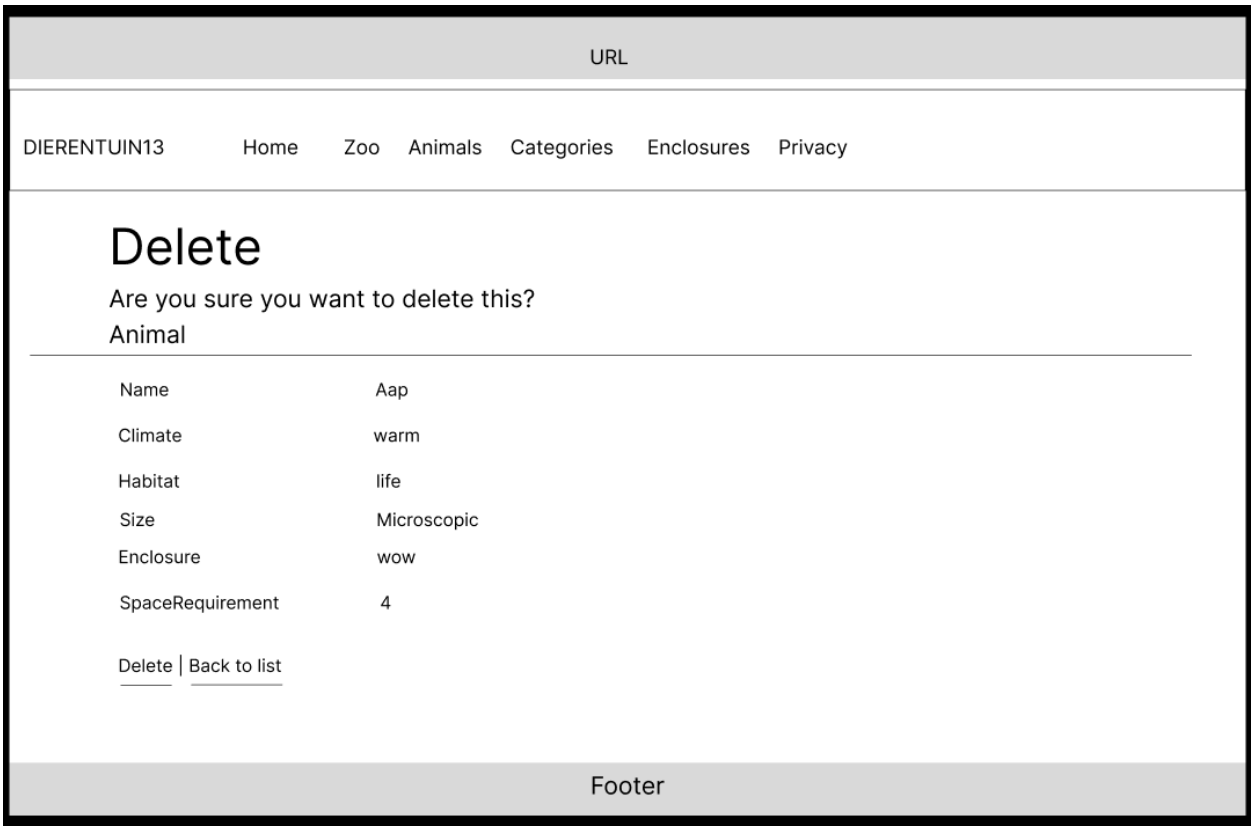

URL
DIERENTUIN13
Home
Animals
Categories
Enclosures
Privacy
Animals
Create new
Find by name, species, categories etc
Search
Sunrise
Sunset
Feedingtime
Name
Species
Categories
Size
Dietaryclass
ActivityPattern
Enclosure
SpaceRequirement
SecurityRequirement
Action
Aap
dier
zoogdier
Microscopic
Carnivore
duirnial
6
low
Edit|Detail|Delete
Footer

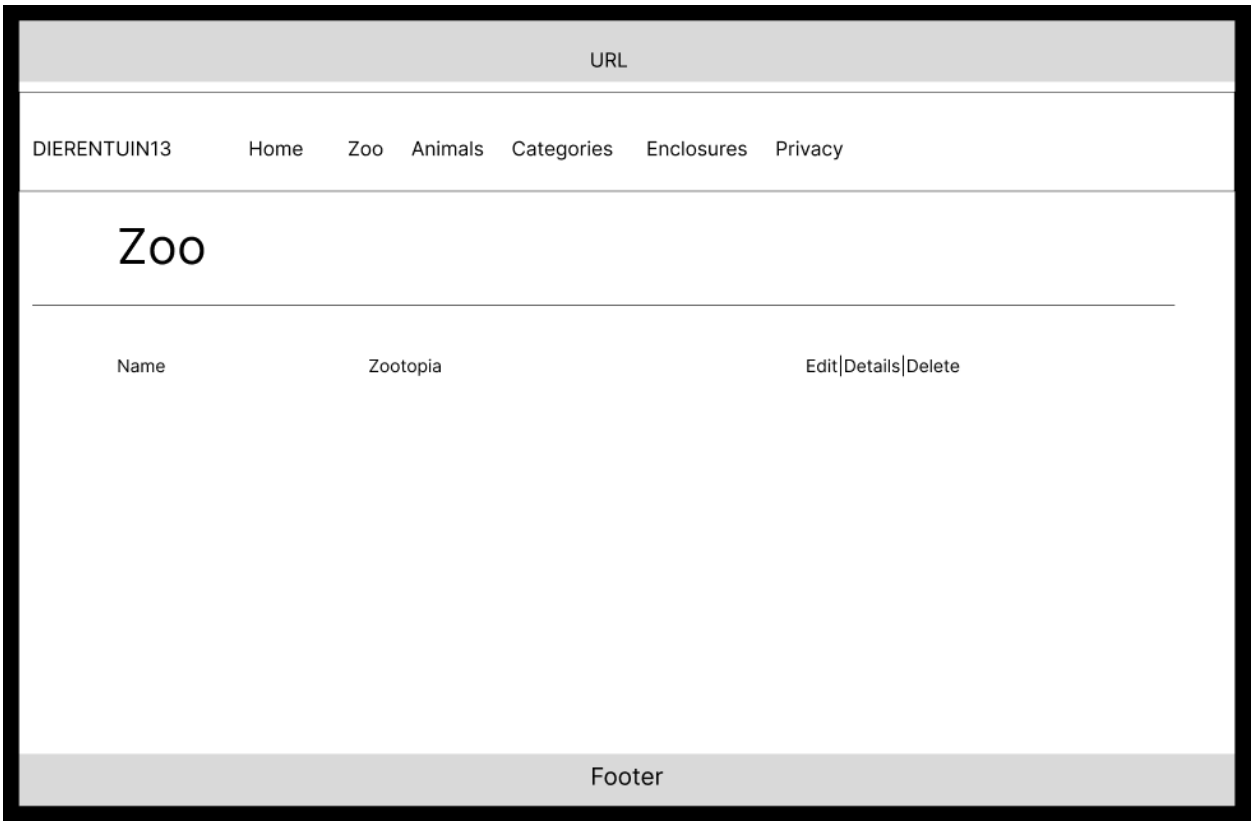

URL
DIERENTUIN13
Home
Zoo
Animals
Categories
Enclosures
Privacy
Categories
Create new
Find by name:
Search
Name
Aap
Edit|Detail|Delete
Footer


URL
DIERENTUIN13
Home
Zoo
Animals
Categories
Enclosures
Privacy
Enclosures
Create new
Find by name, climate, habitat type, security level or size:
Search
Name
Climate
Habitat
SecurityLevel
Size
Aap
Tropical
Forest
Low
3
Edit|Detail|Delete
Footer

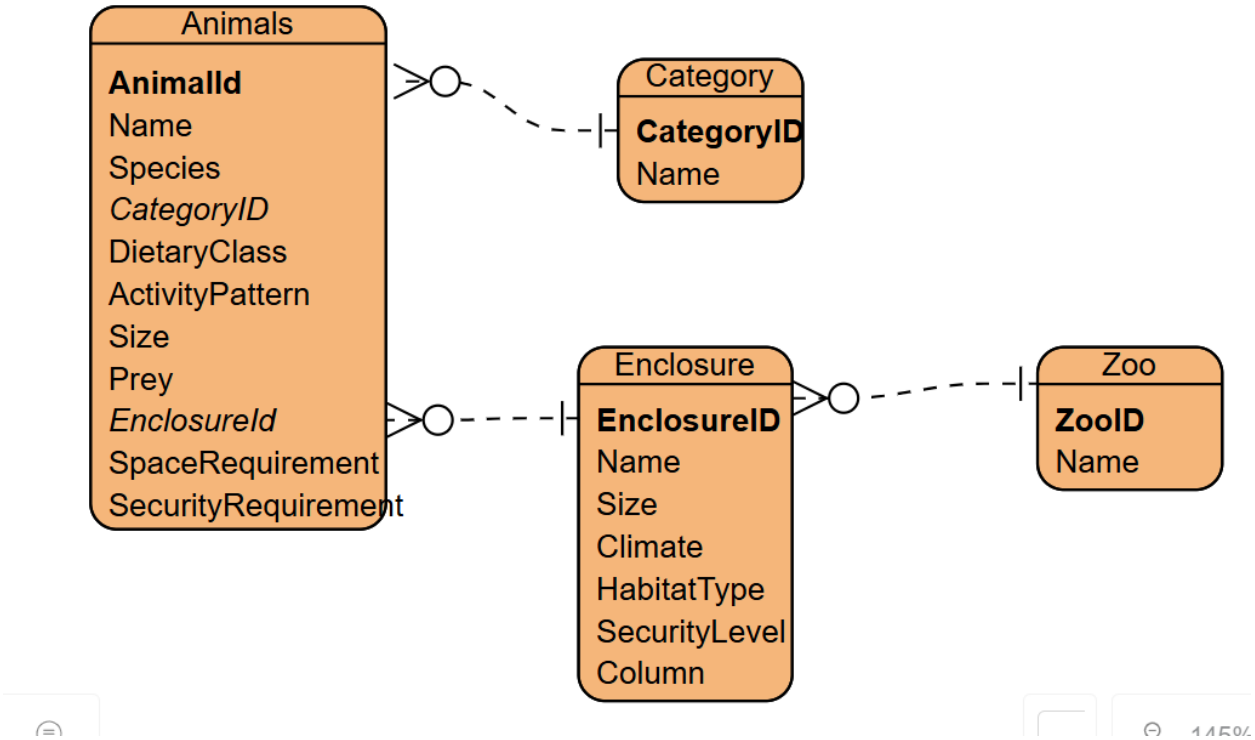

URL
DIERENTUIN13
Home
Zoo
Animals
Categories
Enclosures
Privacy
Edit
Enclosure/categories/animals
Name
Climate
Habitat
Size
Save
Back to list
Footer


URL
DIERENTUIN13
Home
Zoo
Animals
Categories
Enclosures
Privacy
Details
Enclosure/categories/animals
Name
Aap
Climate
warm
Habitat
life
Size
Microscopic
Enclosure
wow
SpaceRequirement
4
Edit | Back to list
Footer


URL
DIERENTUIN13
Home
Zoo
Animals
Categories
Enclosures
Privacy
Delete
Are you sure you want to delete this?
Animal
Name
Aap
Climate
warm
Habitat
life
Size
Microscopic
Enclosure
wow
SpaceRequirement
4
Delete | Back to list
Footer


URL
DIERENTUIN13
Home
Zoo
Animals
Categories
Enclosures
Privacy
Zoo
Name
Zootopia
Edit|Details|Delete
Footer

ERD


Animals
AnimalId
Name
Species
CategoryID
DietaryClass
ActivityPattern
Size
Prey
EnclosureId
SpaceRequirement
SecurityRequirement
Category
CategoryID
Name
Enclosure
EnclosureID
Name
Size
Climate
HabitatType
SecurityLevel
Column
Zoo
ZooID
Name