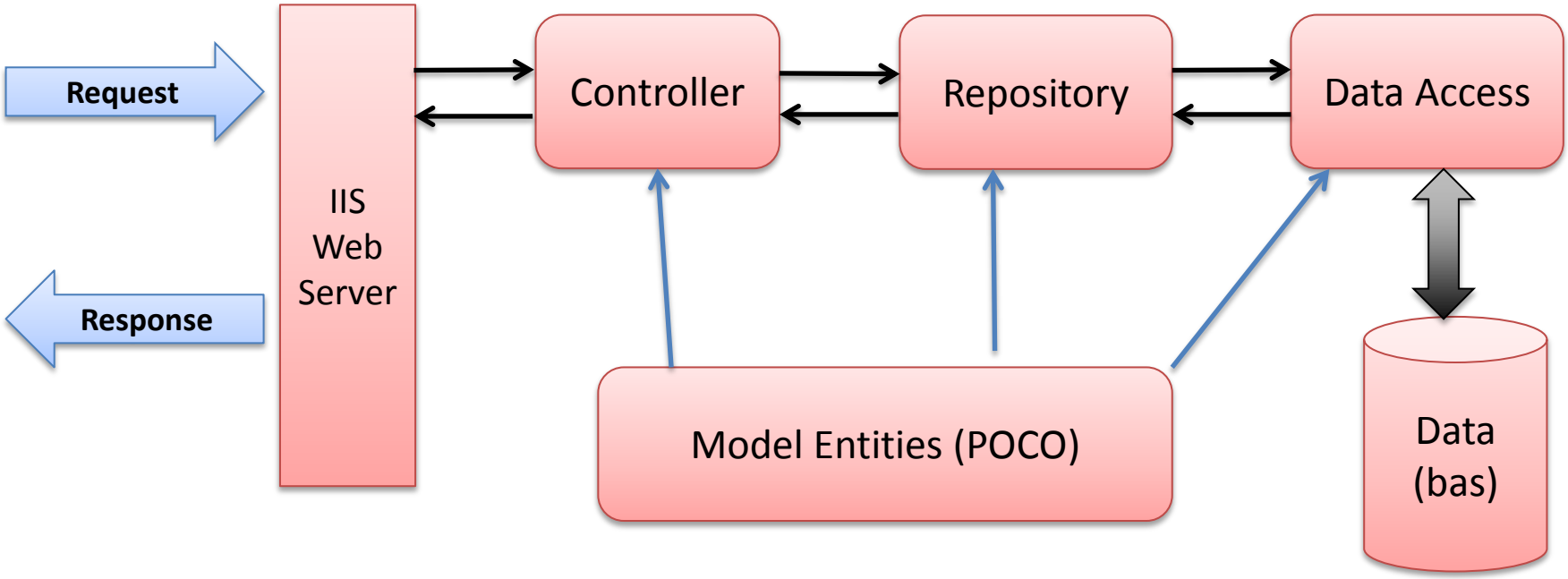
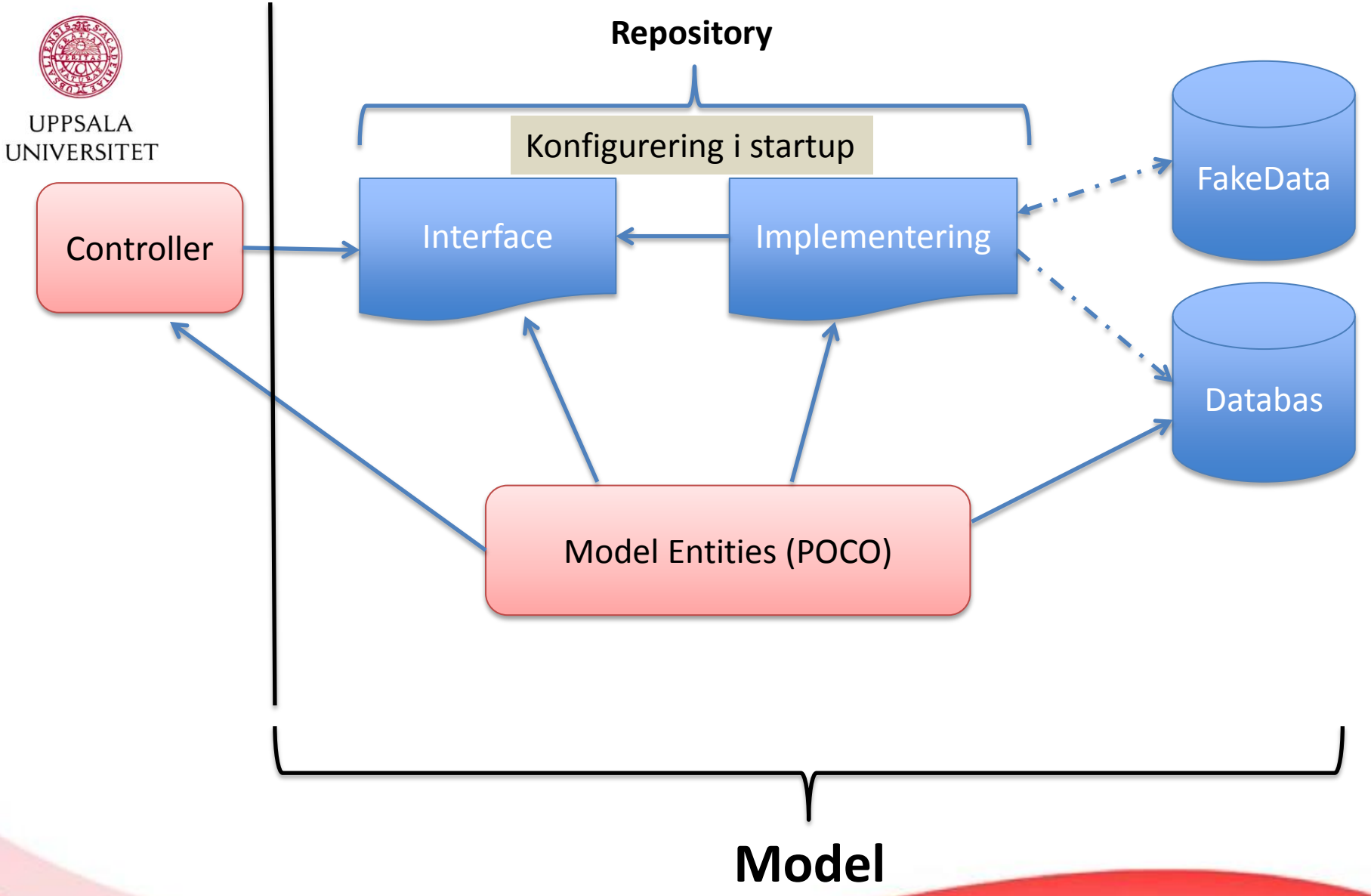
UPPSALA
UNIVERSITET

. NET-programmering

Model

UPPSALA
UNIVERSITET




UPPSALA
UNIVERSITET
Repository
Konfigurering i startup
Controller
Interface
Implementering
FakeData
Databas
Model Entities (POCO)
Model

UPPSALA UNIVERSITET

Model (MVC-mönstret)

- Ansvarar för applikationens data
  - Logiken och reglerna
- Kommunicerar med datalagringen
  - Ex. databas
- Datamodell
  - Data, relationer, semantik och begränsningar

Models

- I mappen Models
  - Klassfiler med koppling till databasen
    - Logik för att söka, lägga till, ta bort, uppdatera
  - Dataklasser som håller i informationen
    - Enkla klasser med properties
    - POCO
    - Ex. Product, Customer, Movie, Book

UPPSALA
UNIVERSITET

POCO (ex.)

```
public class Student
{
    public int StudentID { get; set; }
    public string Code { get; set; }
    public string Name { get; set; }
    public string EnrollmentNo { get; set; }
}
```

Repository

- Kopplingen till databasen
  - Interfaces med en samling att returneras
- Falsk behållare med hårdkodad data att använda tills man kopplar till databasen
  - Ett sätt att se att allt fungerar innan man kör skarpt
  - En klass som implementerar interface

UPPSALA
UNIVERSITET

Interface Repository (ex)

```
public interface ISchoolRepository
{
        IQueryable<Student> Students { get; }
}
```

UPPSALA
UNIVERSITET

Falsk databas (ex)

```
public class FakeSchoolRepository: ISchoolRepository
{
  public IQueryable<Student> Students => new List<Student>
  {
     new Student { StudentID = 1, Code = "L0001",
         Name = "Amelia Gundersson",
          EnrollmentNo = "201804150001" },
     new Student { StudentID = 2, Code = "L0002",
         Name = "Charlie Jansson",
          EnrollmentNo = "201804150002" },
     new Student { StudentID = 3, Code = "L0003",
         Name = "Bertil Paulsson",
        EnrollmentNo = "201804150003" }
   }.AsQueryable<Student>();
}
```

UPPSALA UNIVERSITET

Repository service

- Skapa repository service
  - Lös koppling via services
  - Kontroller kan hämta objekt från interface utan att bry sig om vilken klass som egentligen gör jobbet

- Kod läggs i ConfigureService i StartUp-filen

```
services.AddTransient<ISchoolRepository, FakeSchoolRepository>();
```

Obs! Vi måste lägga till using MySchool.Models i Startup-filen

UPPSALA
UNIVERSITET

Controll

- Kontrollen som ska hantera koppling till model måste få en privat instans av interfacet
  - Ex. `private ISchoolRepository repository;`
- En konstruktor som skapar objektet
  - Ex.

```
public HomeController (ISchoolRepository repo)
{
    repository = repo;
}
```

- Action metod för att hämta datat via objektet och skicka till view
  - Ex.
    public ViewResult Index() => View(repository.Students);

Dependency injection

- Kontroll
  - Privat instans av interface
  - Konstruktor som skapar objektet
  - Action som skickar med model-klassen till vyn
- ConfigureService
  - Kod som kopplar interface med implementationen

Lägga till model

- I ViewImports.cshtml
  - `@using MySchool.Models`

UPPSALA UNIVERSITET

View

- I vyn läggs model till
  - @model *modelnamn*
    - *Ex.* `@model IEnumerable<Student>`
  - Och vi kan via model-objektet loopa igenom och visa upp innehållet ex. via foreach

```
@foreach (Student student in Model)
{
    <div>
      <p>@student.StudentId</p>
      <p>@student.Code</p>
      <p>@student.Name</p>
      <p>@student.EnrollmentNo</p>
    </div>
}
```

UPPSALA
UNIVERSITET

Kodgenomgång