

The background of the slide is a close-up, artistic photograph of a large stack of colorful folders or files. The folders are in various colors, including red, orange, yellow, green, and blue, and are arranged in a way that creates a sense of depth and texture. The text "Data Modelling and Persistence" is overlaid in the center of the image in a white, sans-serif font.

Data Modelling and Persistence

Modelling data with POJOs

```
public class Venue {  
    private long id;  
    private String name;  
    private int capacity;  
    public Venue() {  
    }  
    // Getters and setters omitted for brevity  
}
```

Annotate POJOs for persistence

```
@Entity
@Table(name = "venues")
public class Venue {

    private long id;

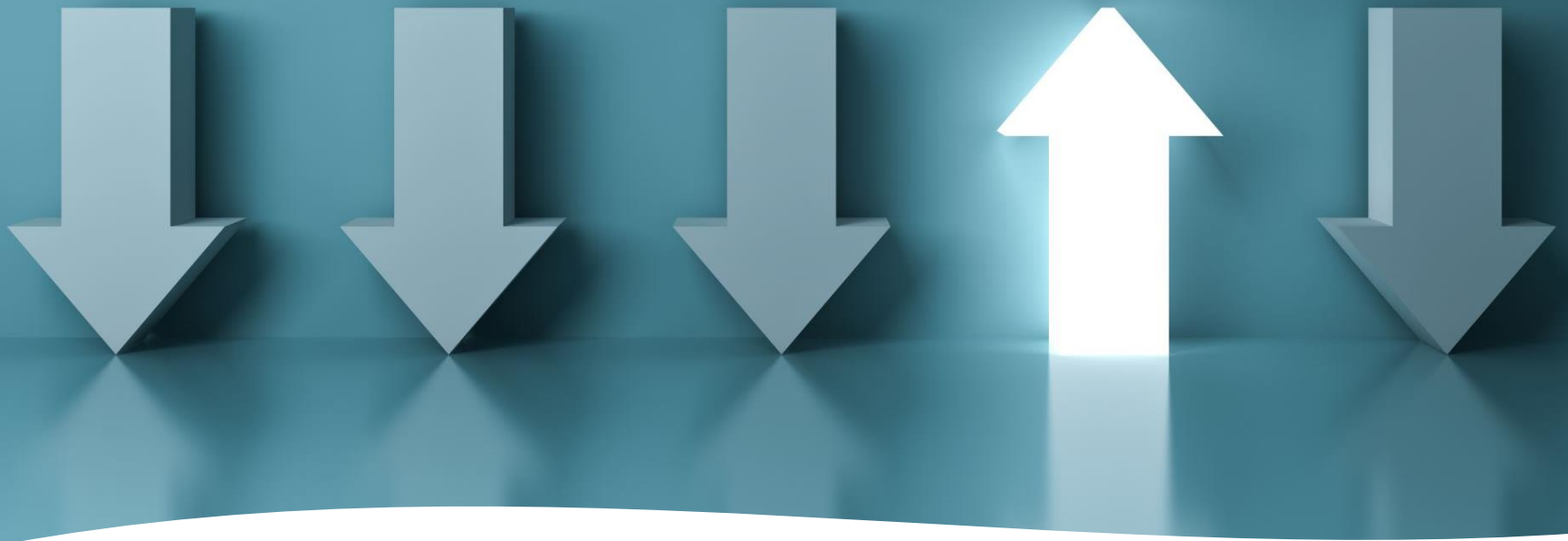
    private String name;

    private int capacity;

    public Venue() {

    }

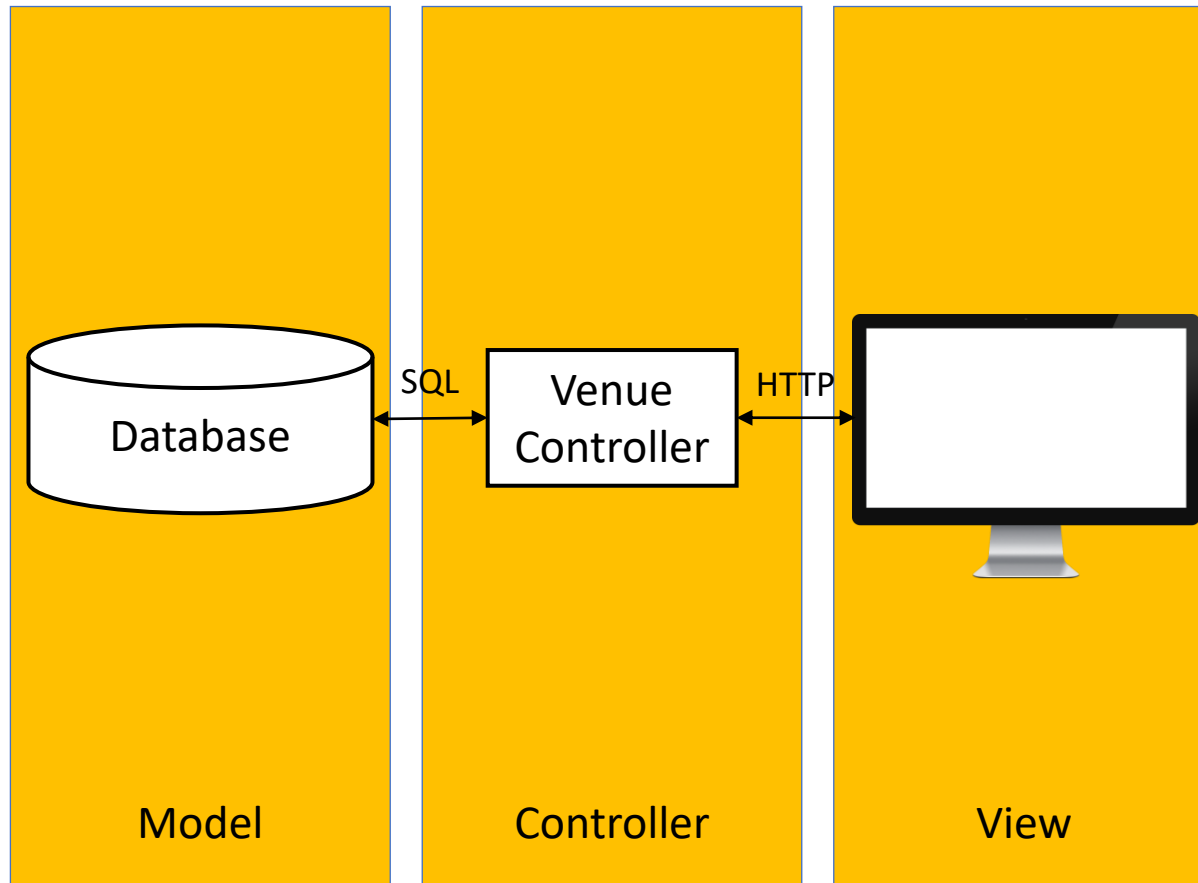
    // Getters and setters omitted for brevity
}
```



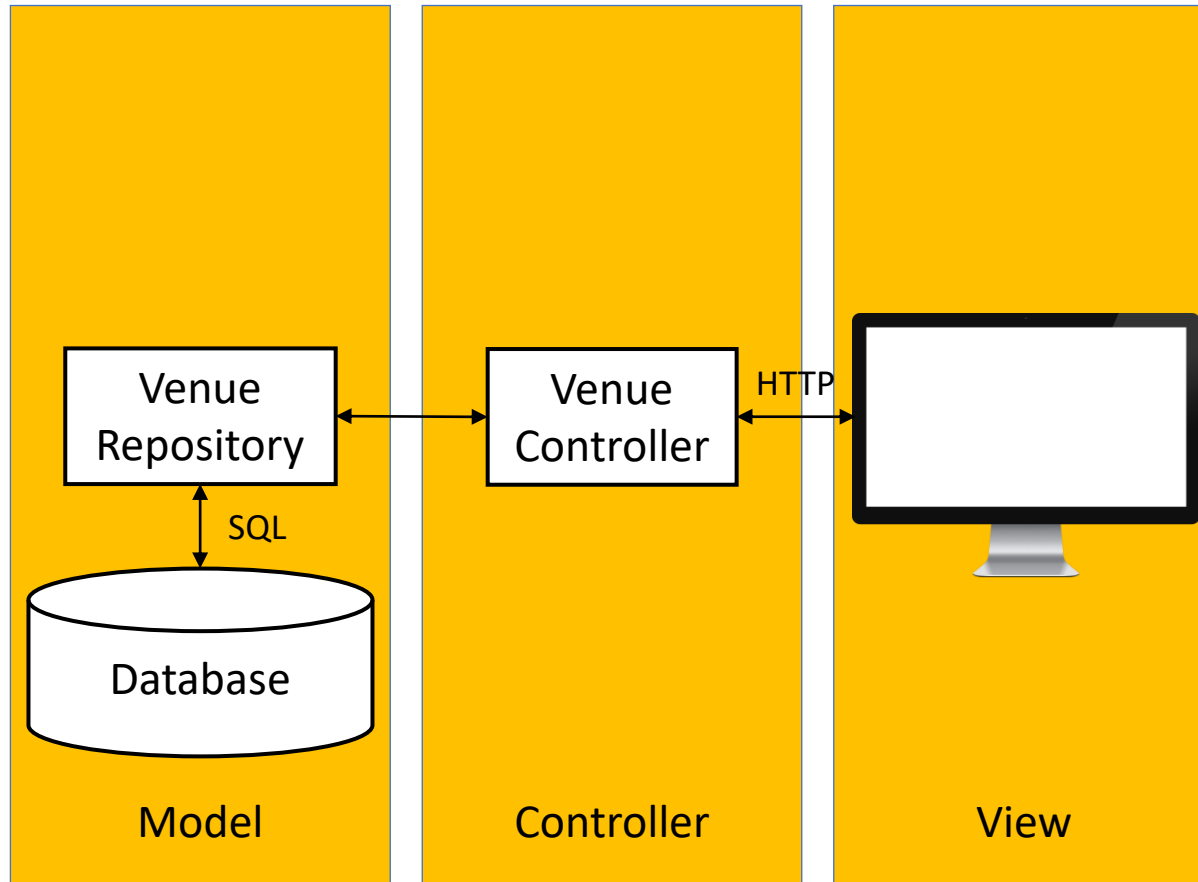
Data relationships

- @OneToMany
 - “A venue can host many events”
- @ManyToOne
 - “An event has one venue”
- @OneToOne
 - “A venue has a manager”

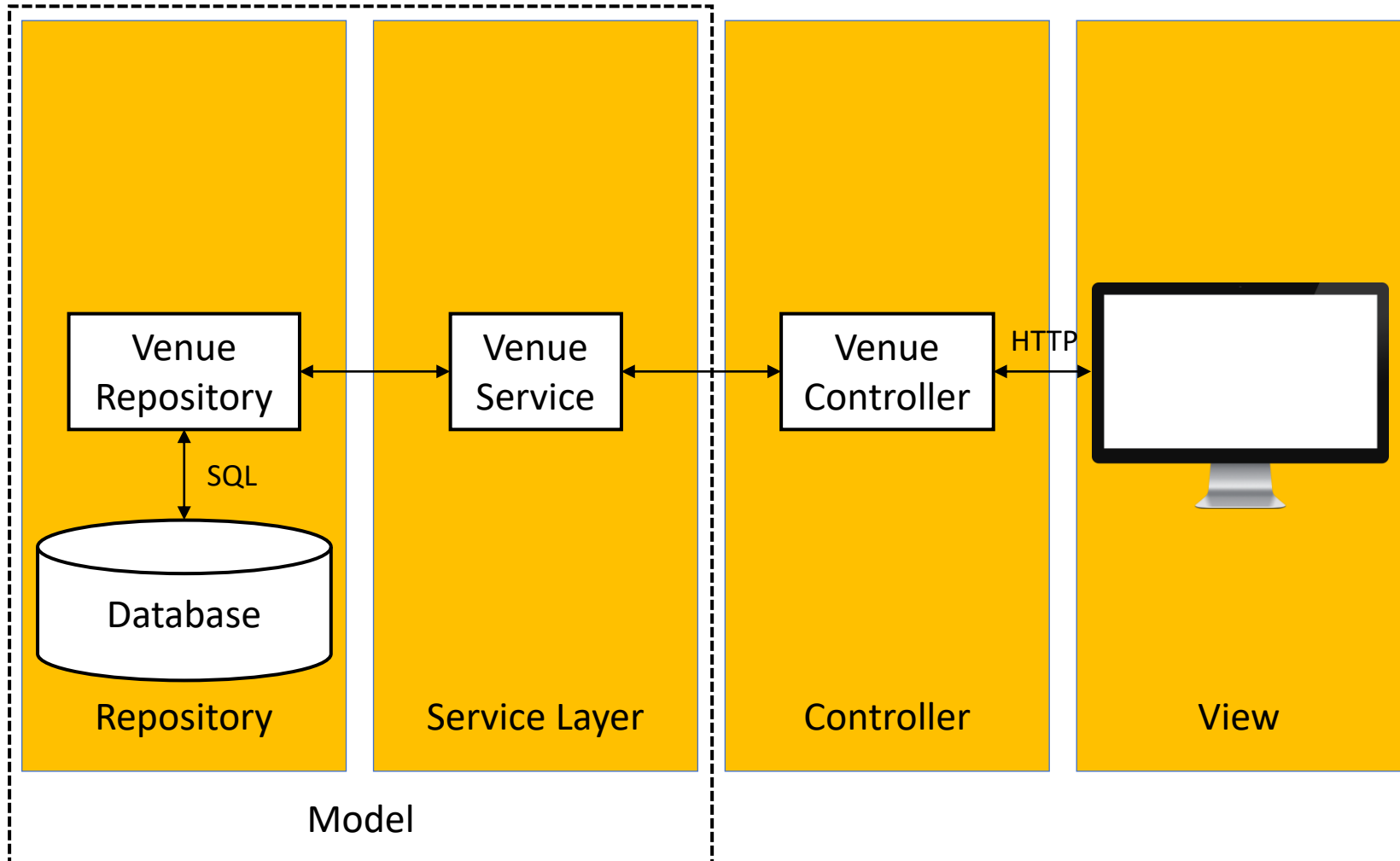
Data Access Architecture



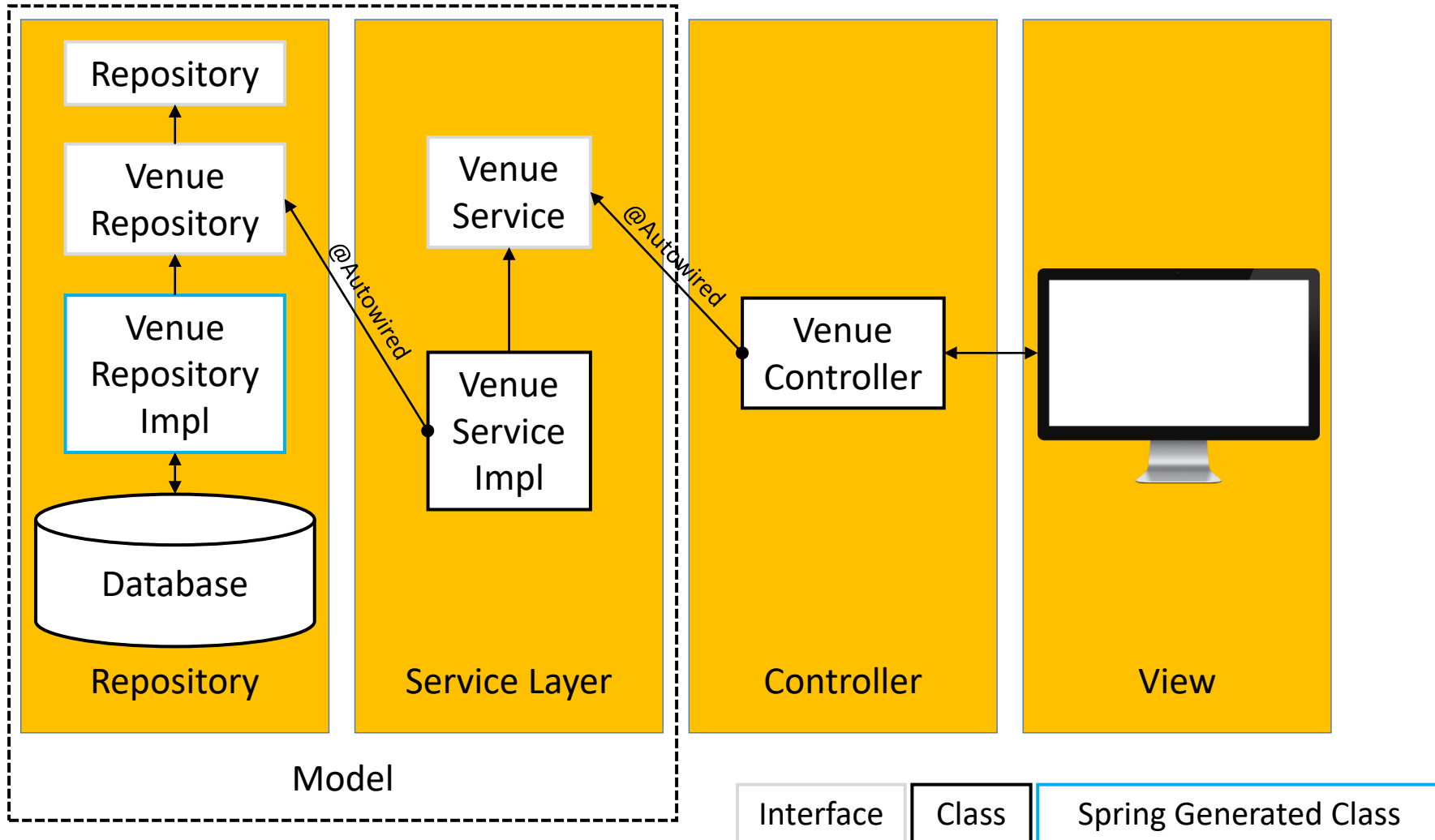
Data Access Architecture



Spring's Data Access Architecture



Spring's Data Access Objects



Spring repositories

```
public interface VenueRepository extends CrudRepository<Venue, Long> {  
      
      
}
```

The default implementation provides:

- count()
- findOne(long id), findAll()
- save(Venue venue), delete(long id)
- More...

More Spring repository magic

Simply define the queries you want as methods in your repository interface:

```
public interface VenueRepository extends CrudRepository<Venue, Long> {  
    public Iterable<Venue> findAllByName(String name);  
    public Iterable<Venue> findAllByNameOrderByNameAsc(String name);  
    public Venue findFirstByNameOrderByNameAsc(String name);  
  
    public Venue findByNameContainingAndCapacity(String nameSearch, int capacity);  
  
    public Iterable<Venue> findAllByCapacityBetween(int min, int max);  
}
```

Querying data via a Service interface

Create a Service interface to expose the methods you need and get Spring to auto-wire the repository:

```
@Service
public class VenueServiceImpl implements VenueService {

    @Autowired
    private VenueRepository venueRepository;

    @Override
    public Iterable<Venue> findAll() {
        return venueRepository.findAll();
    }
}
```