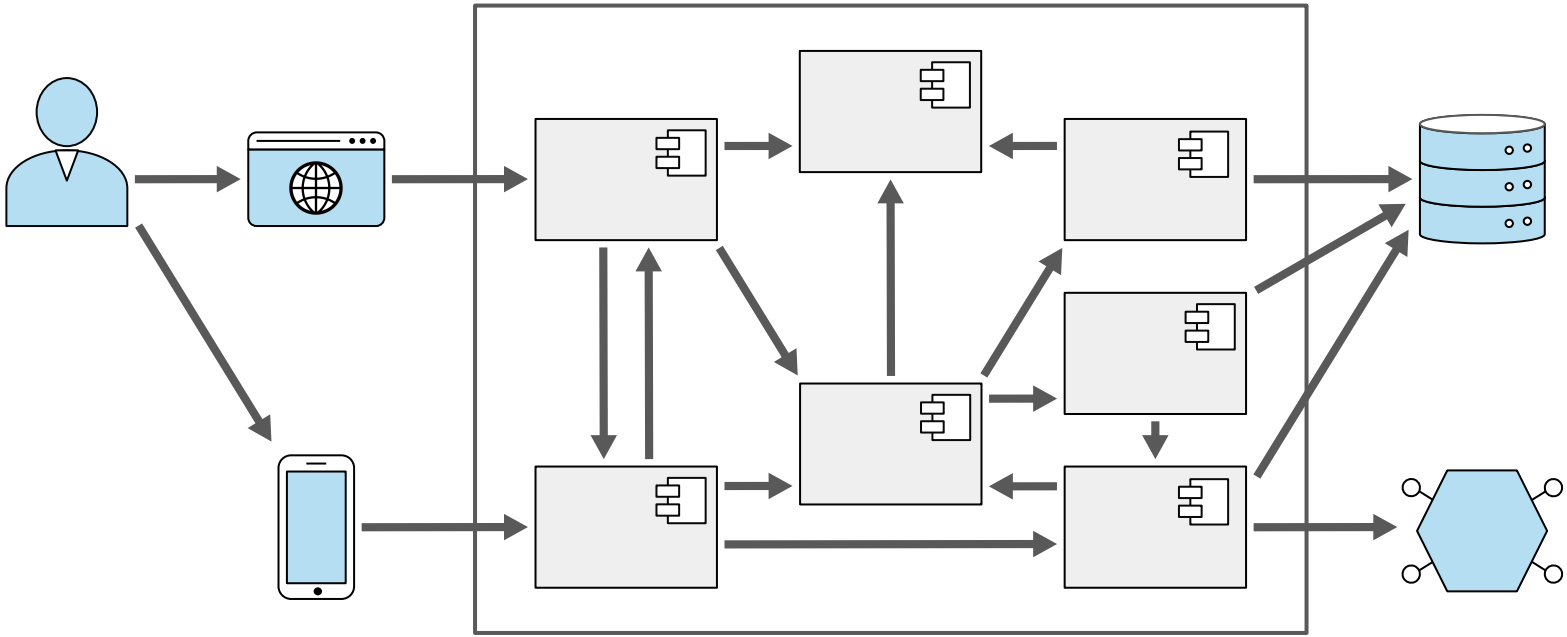
HappyCoders.eu

Hexagonal Architecture: Robust Software With Interfaces Instead of Layers

Sven Woltmann

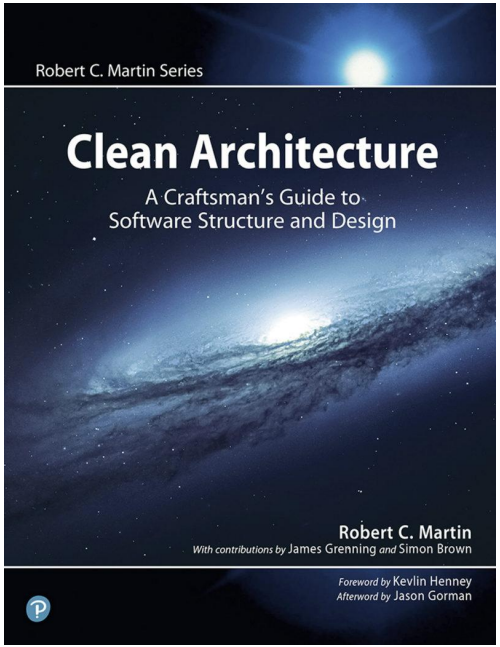
What is Software Architecture?



What Is the Goal of Software Architecture?

“The goal of a software architecture is to minimize the human resources required to build and maintain the required system.”

“If that effort is low, and stays low throughout the lifetime of the system, the design is good.”

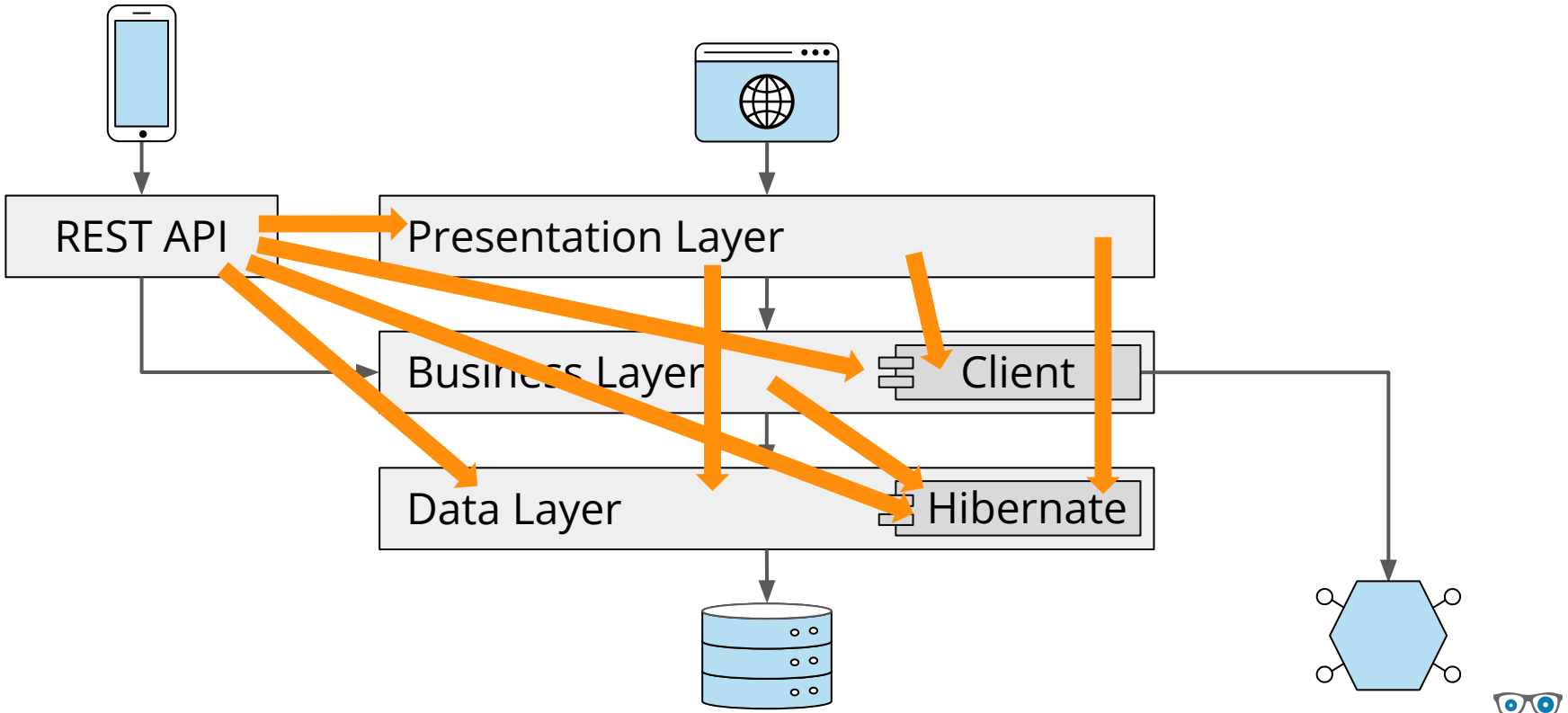


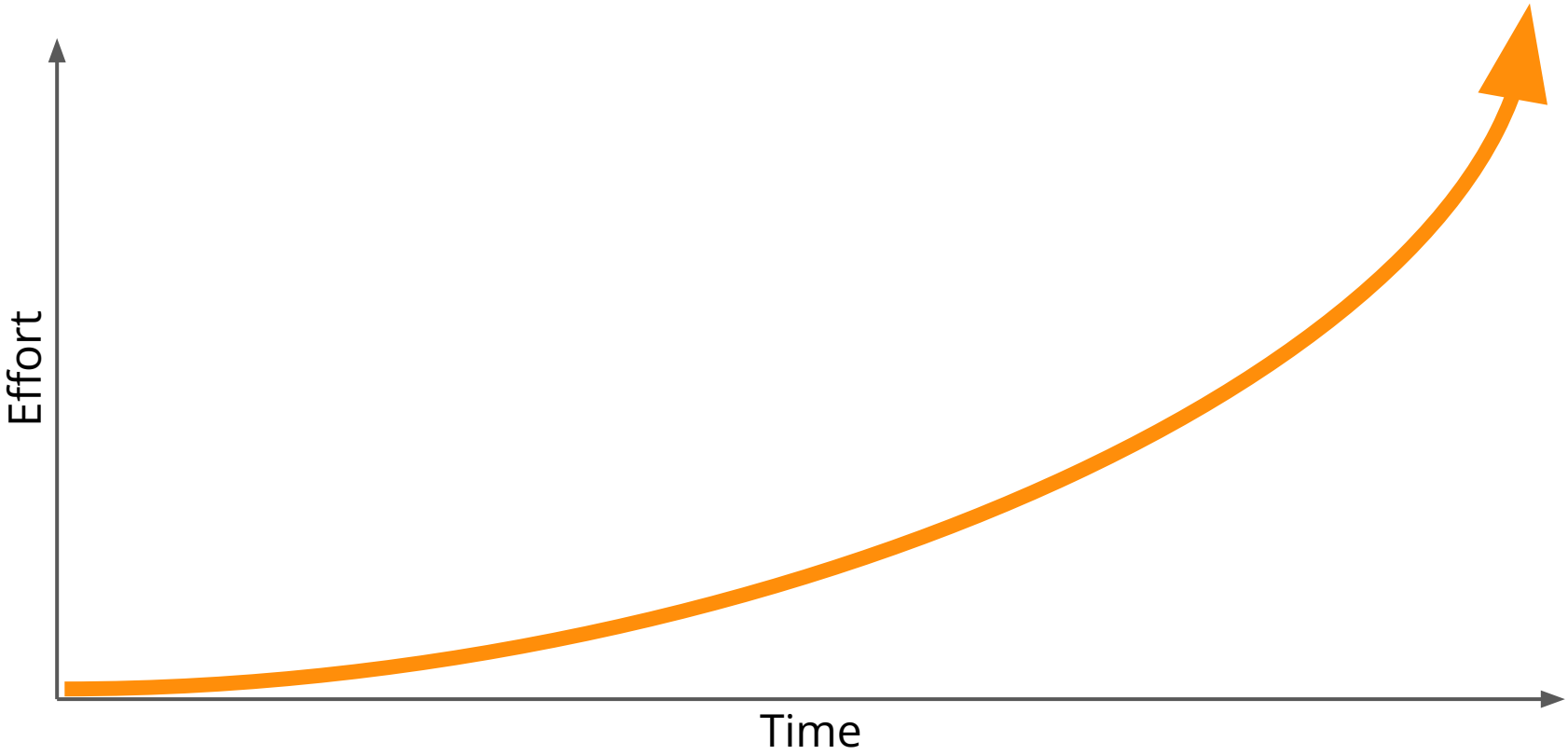
What Is the Goal of Software Architecture?

A good architecture makes it possible to change software during its lifetime with as little effort as possible.

“To keep software soft.”
— Robert C. Martin

Layered Architecture



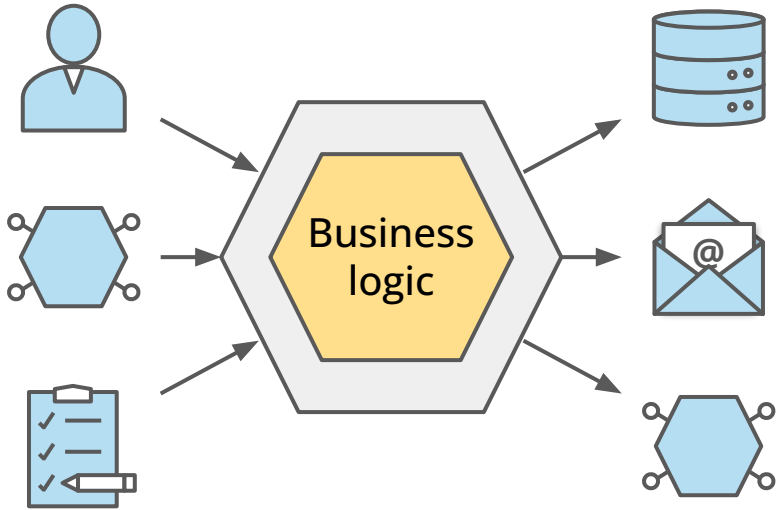


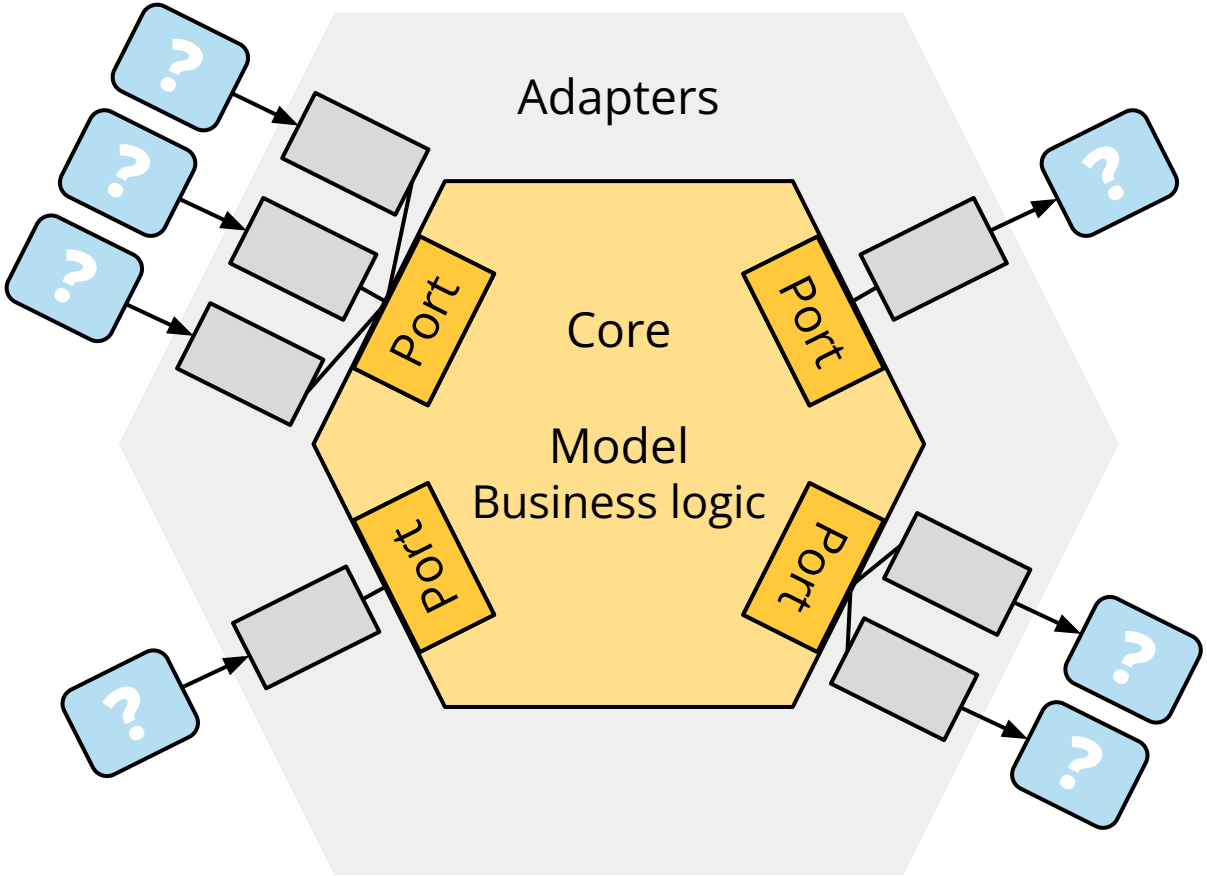
Hexagonal Architecture

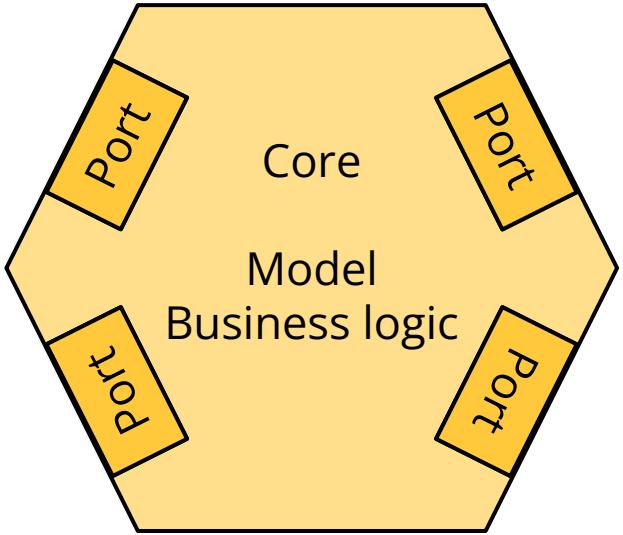
https://alistair.cockburn.us/hexagonal-architecture/

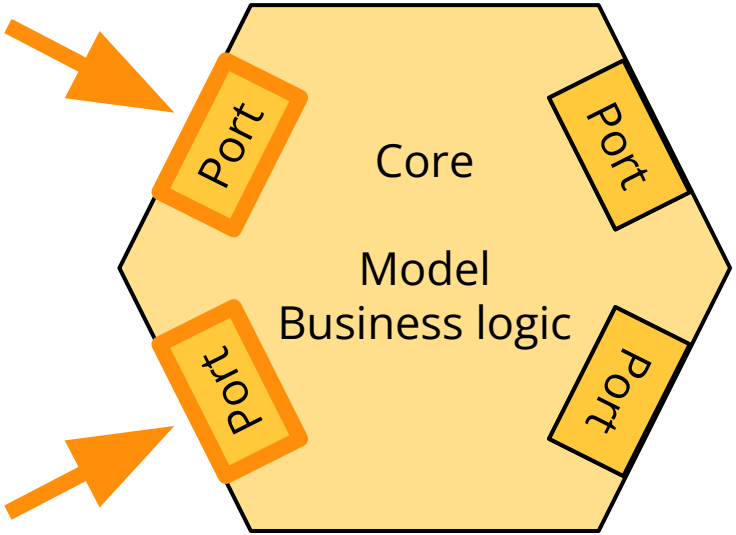
Goals

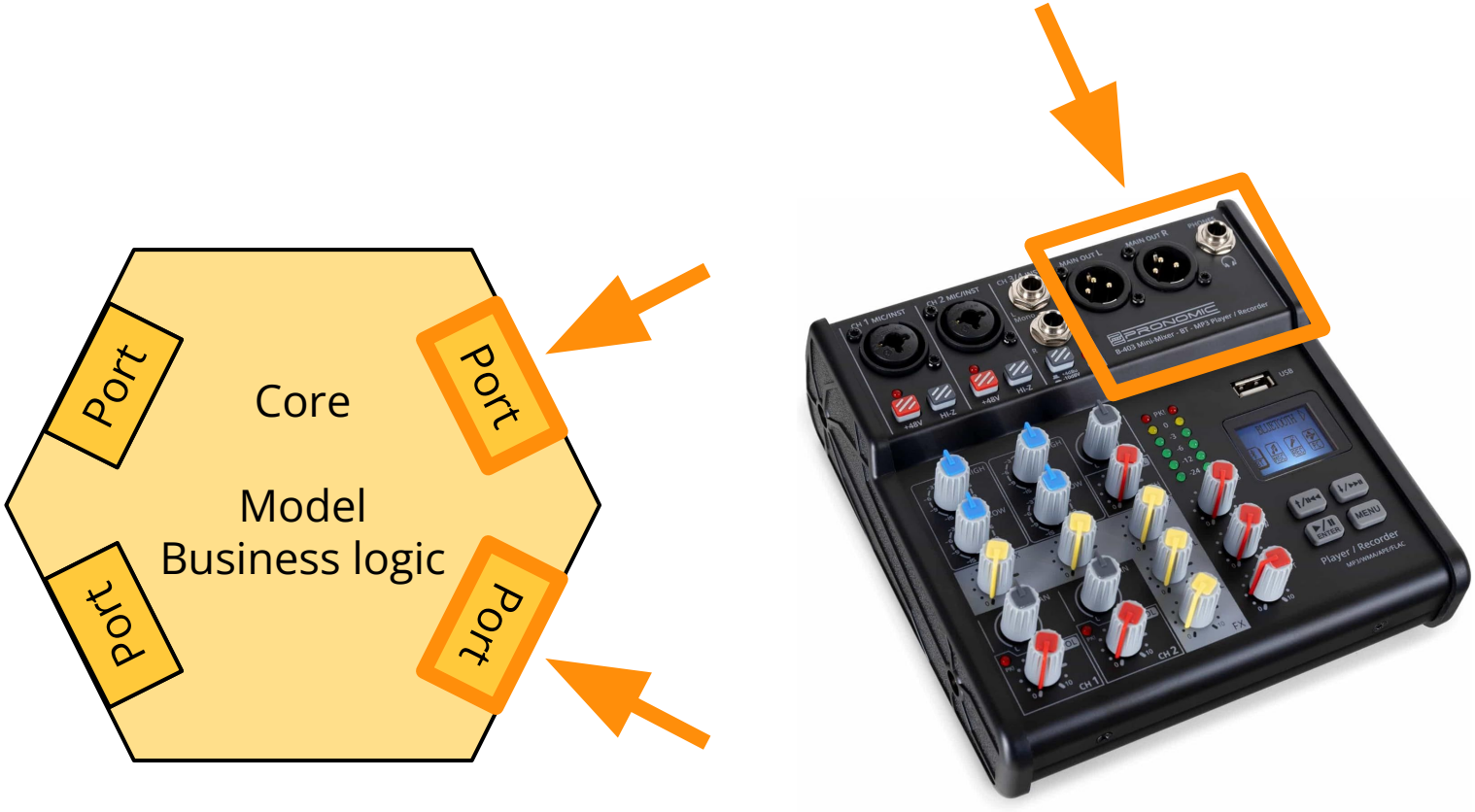
1. Isolation from *controlling* infrastructure
2. Isolation from *controlled* infrastructure
3. Modernization of infrastructure without adapting the business logic

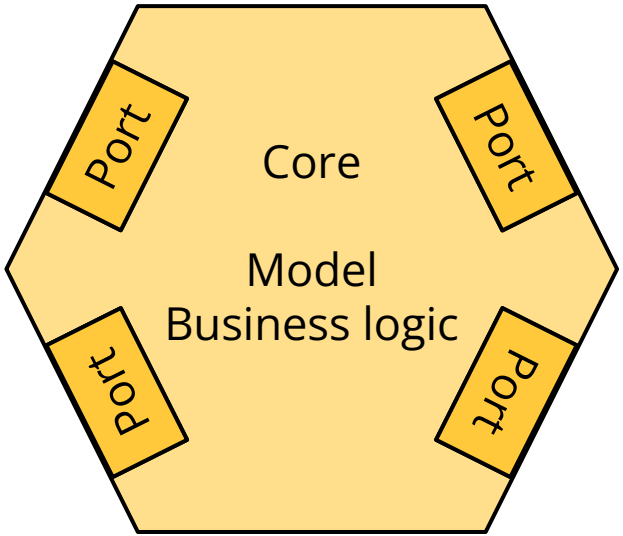


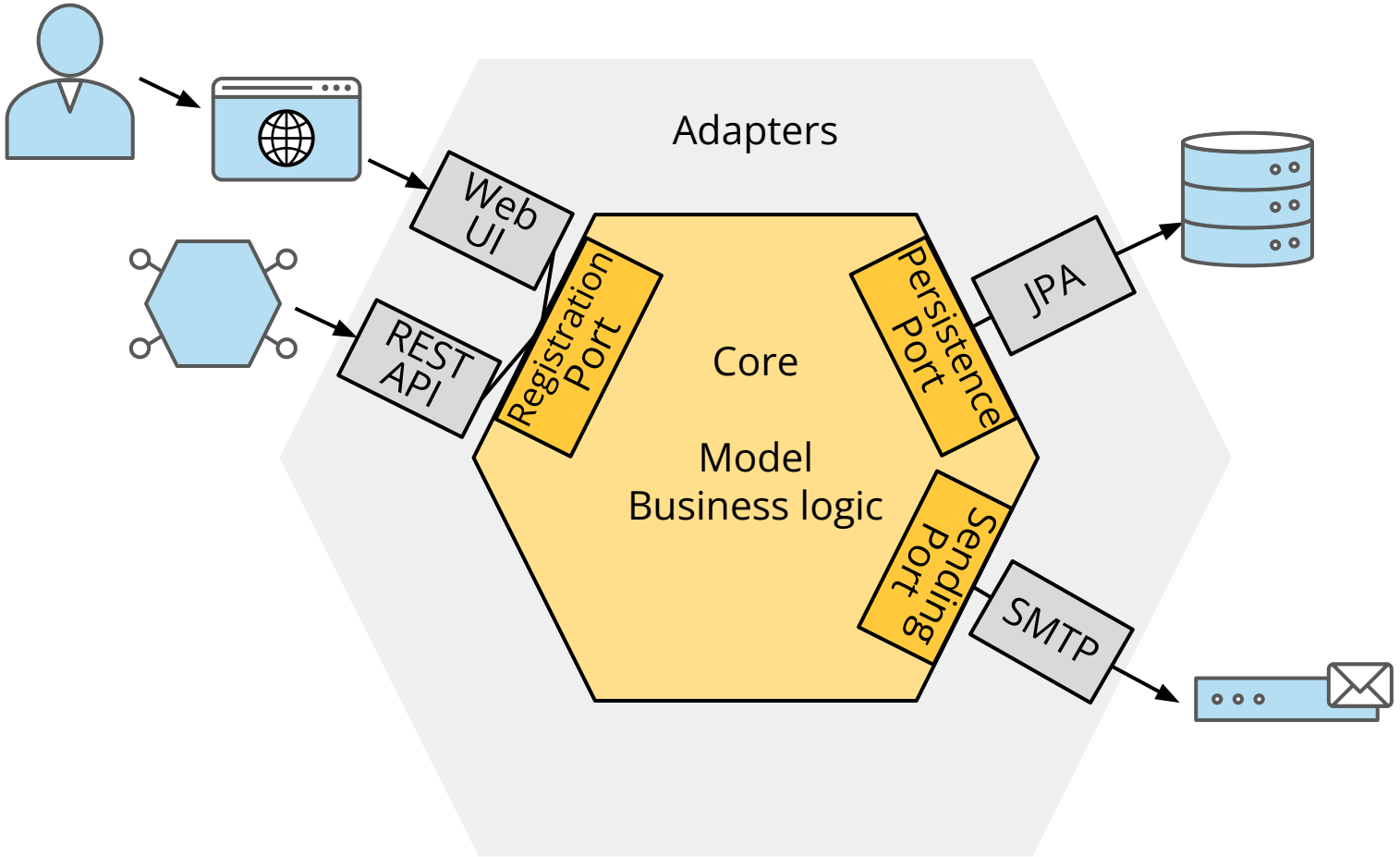


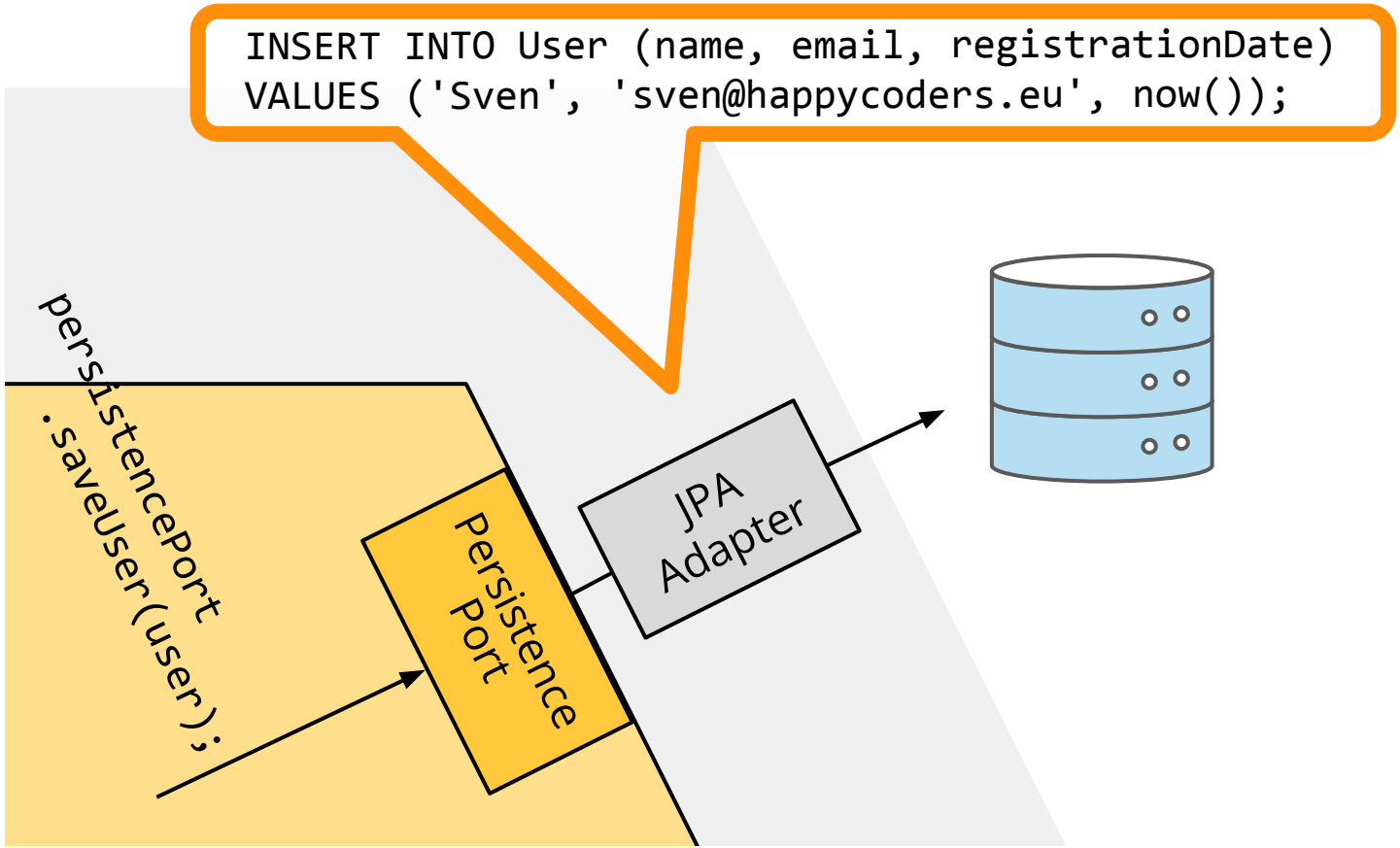


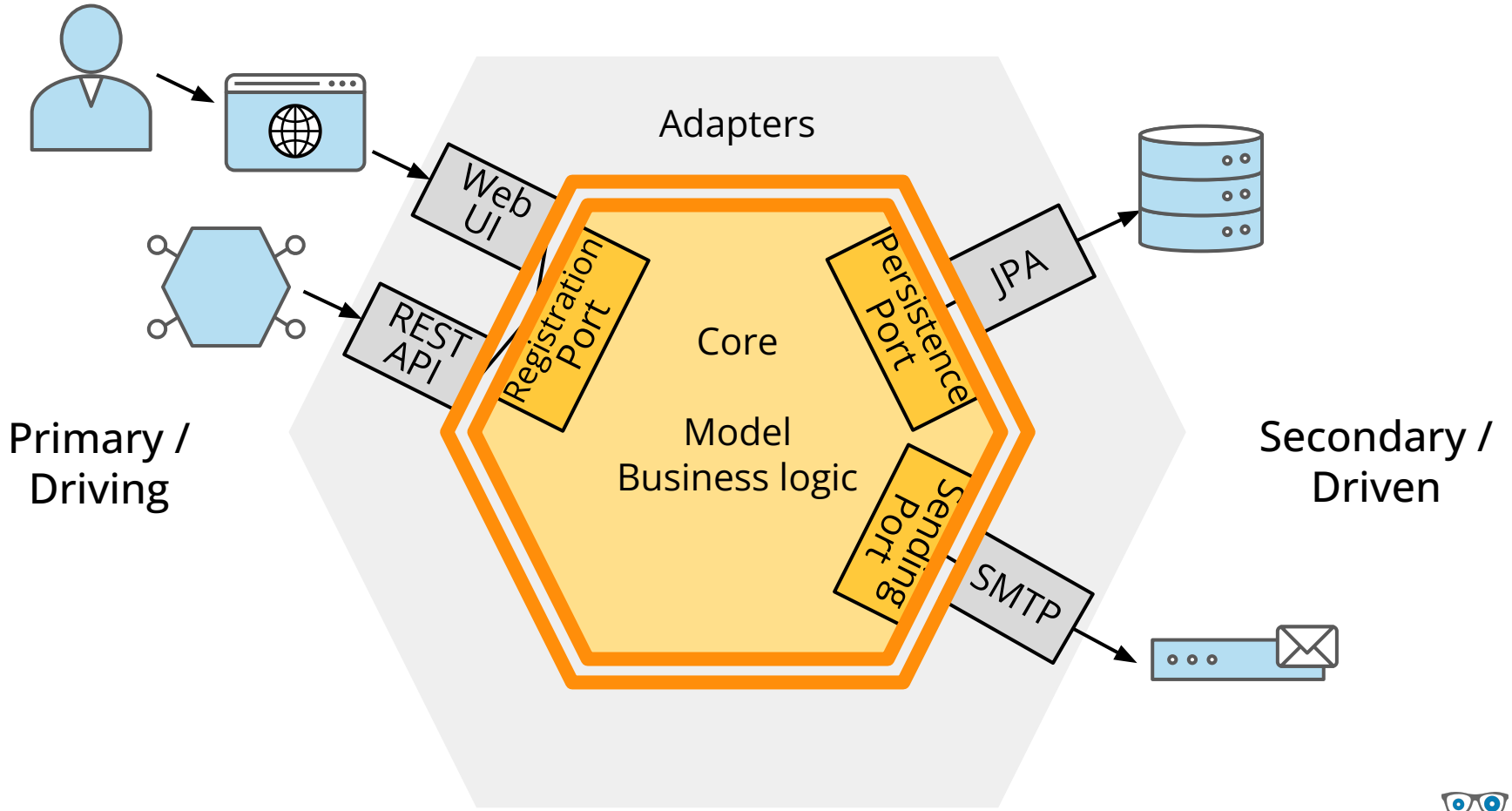


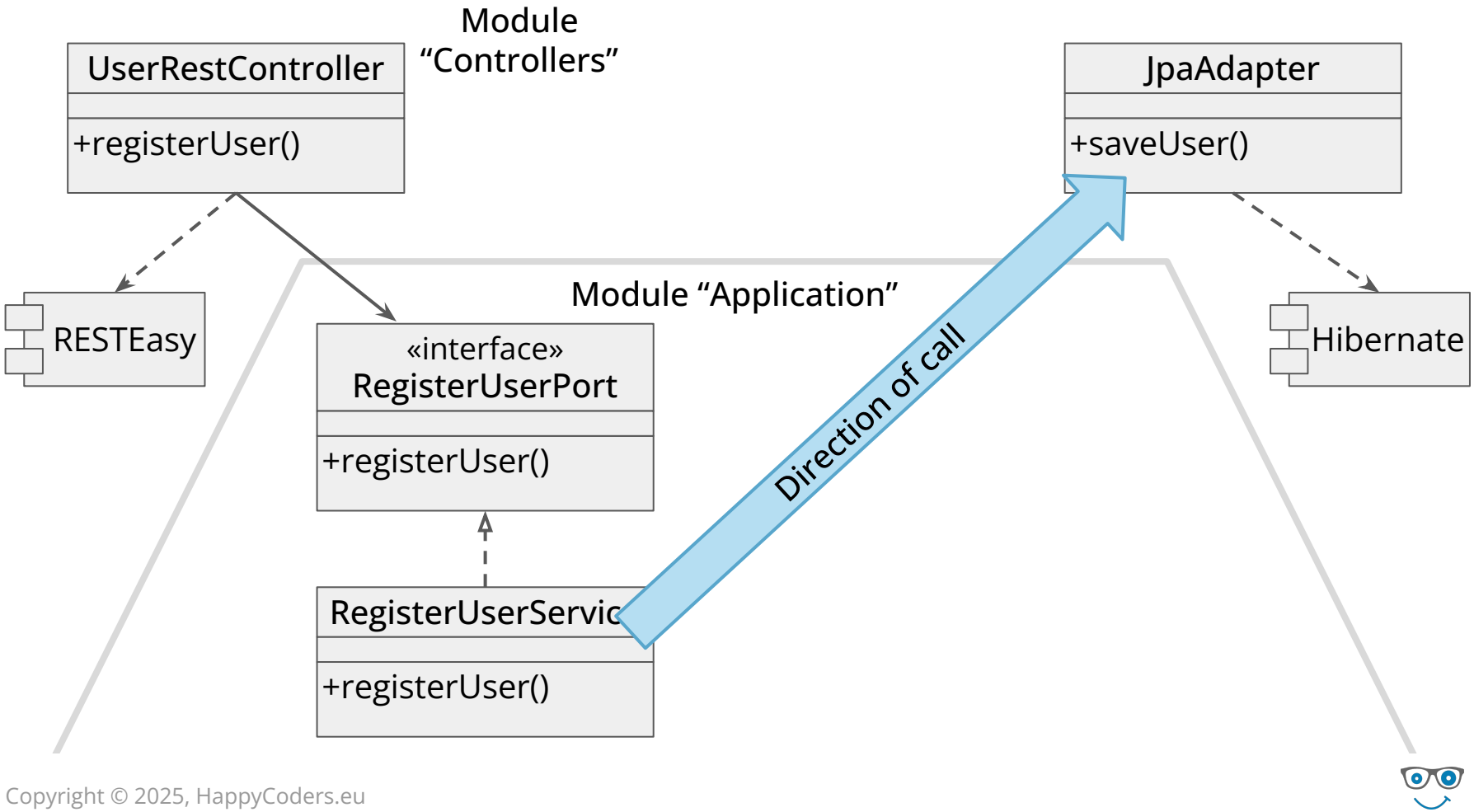






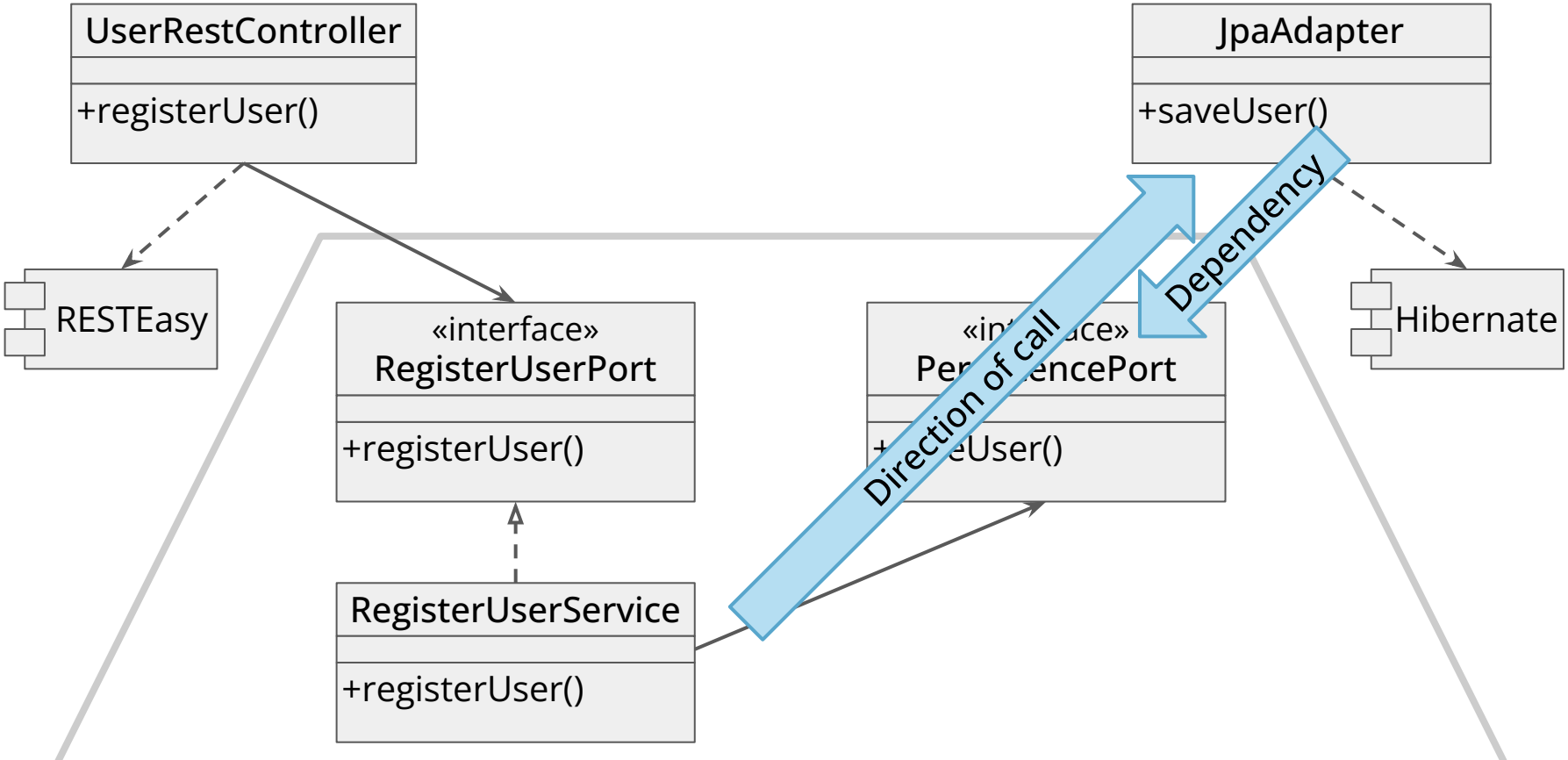


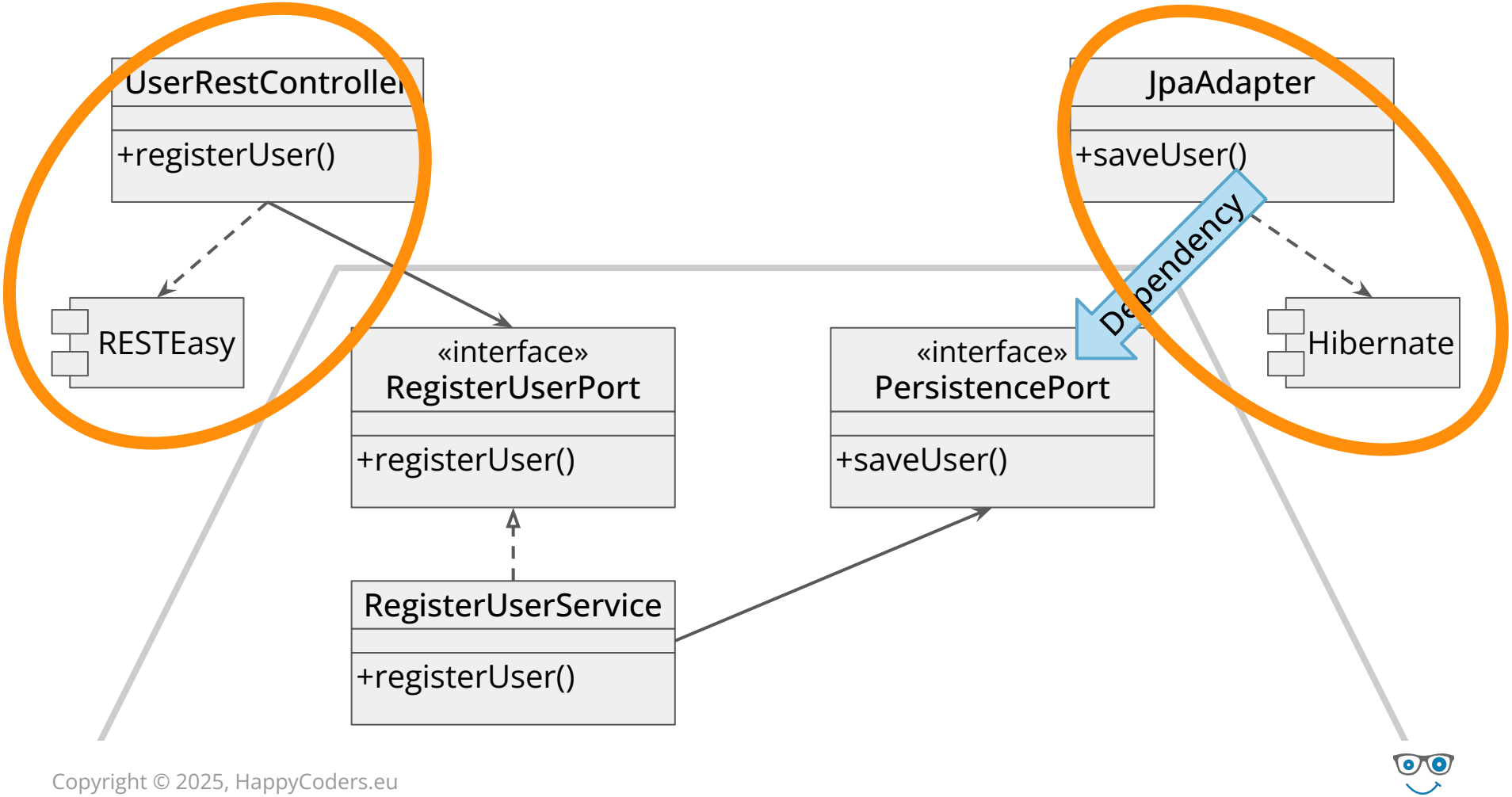




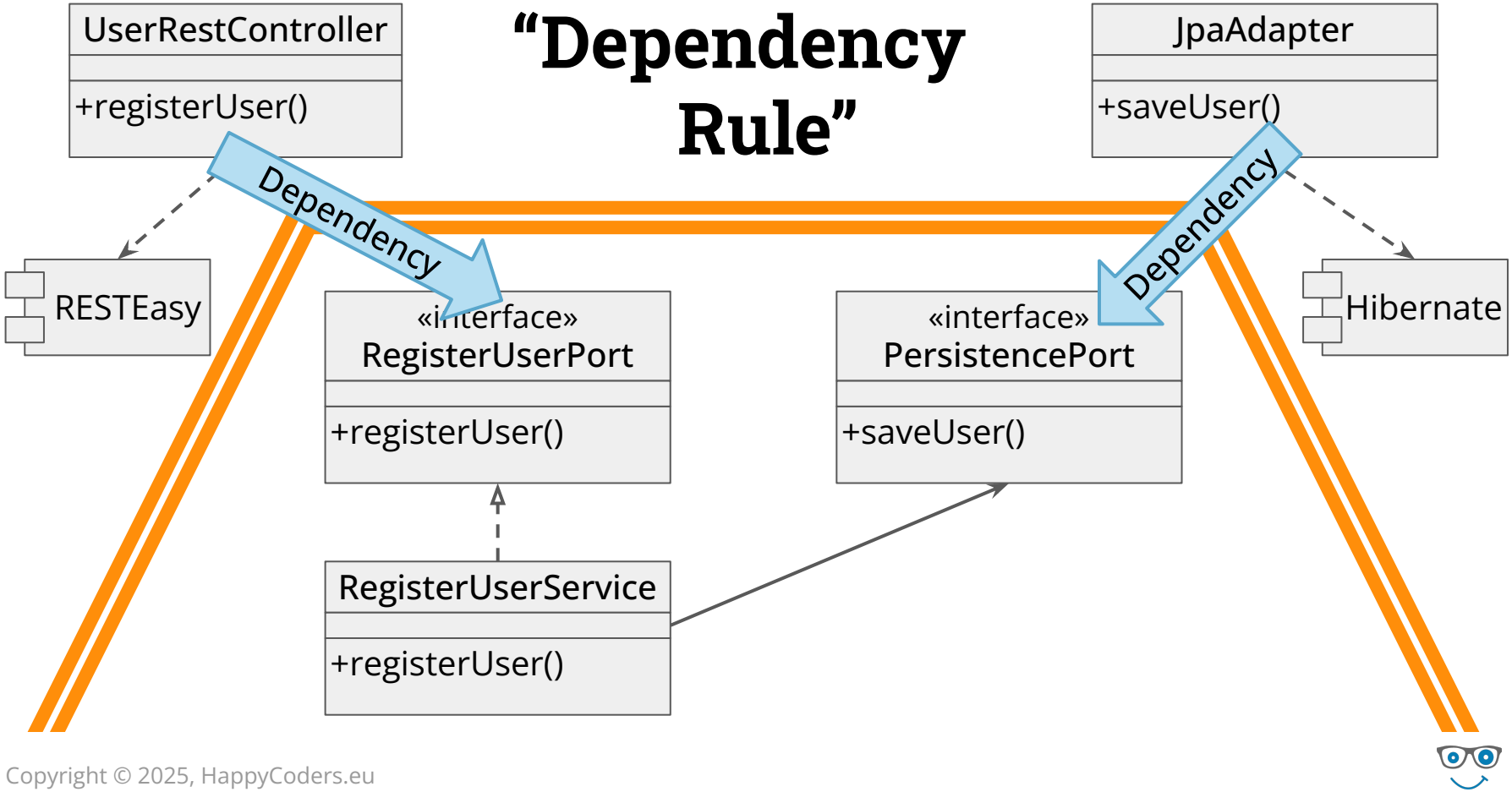
Dependency Inversion Principle

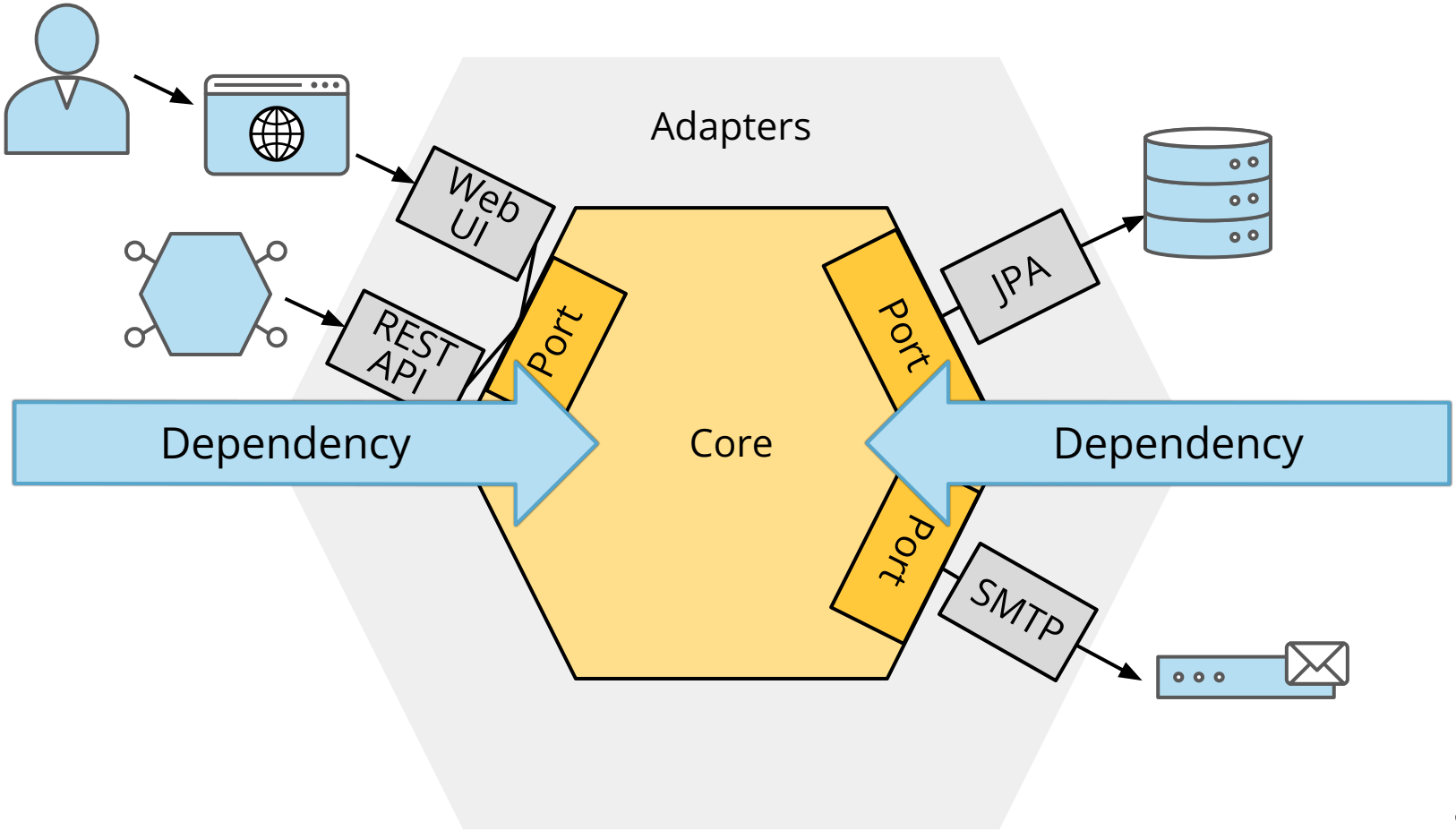
SOLID





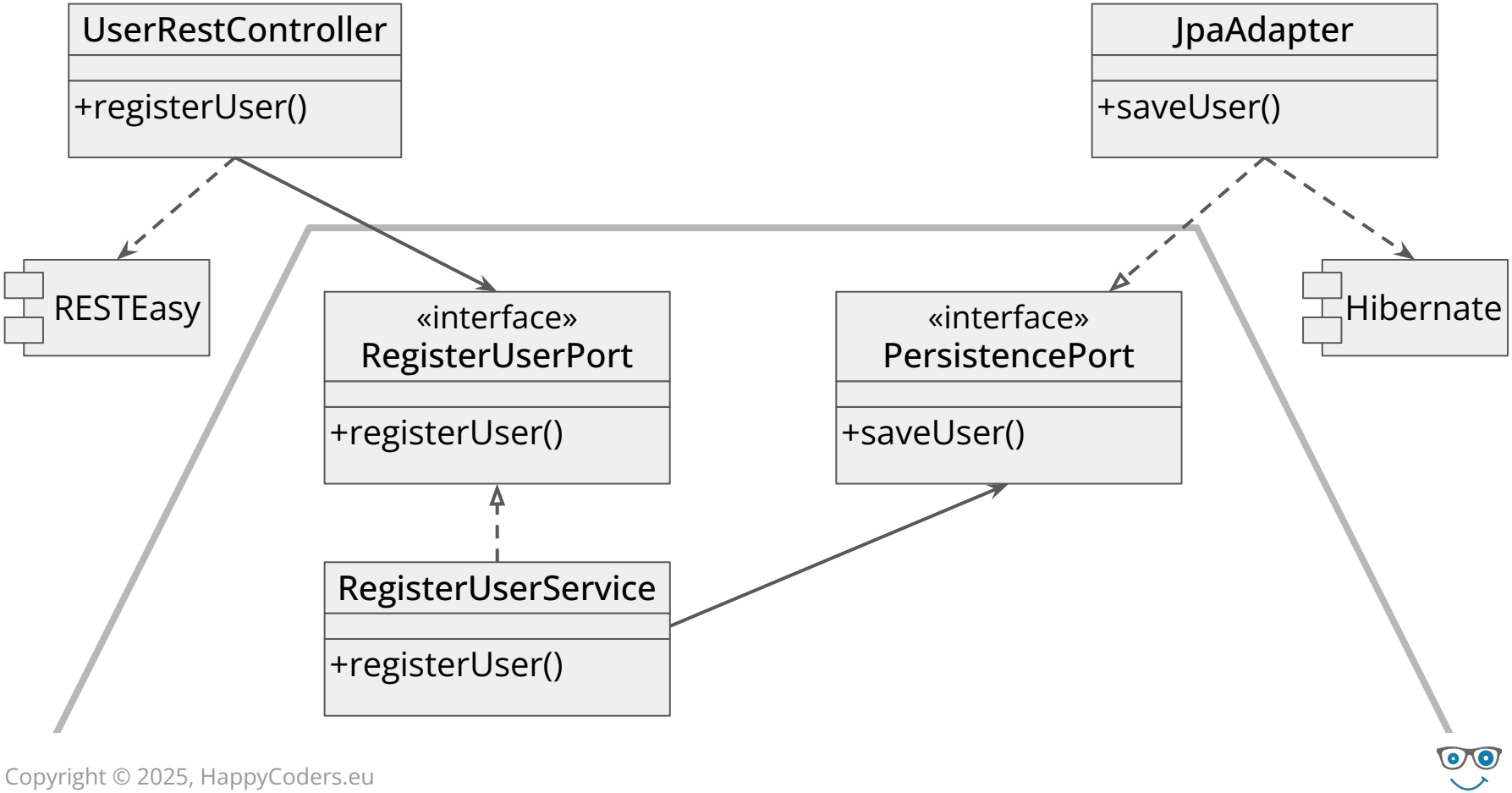
“Dependency Rule”

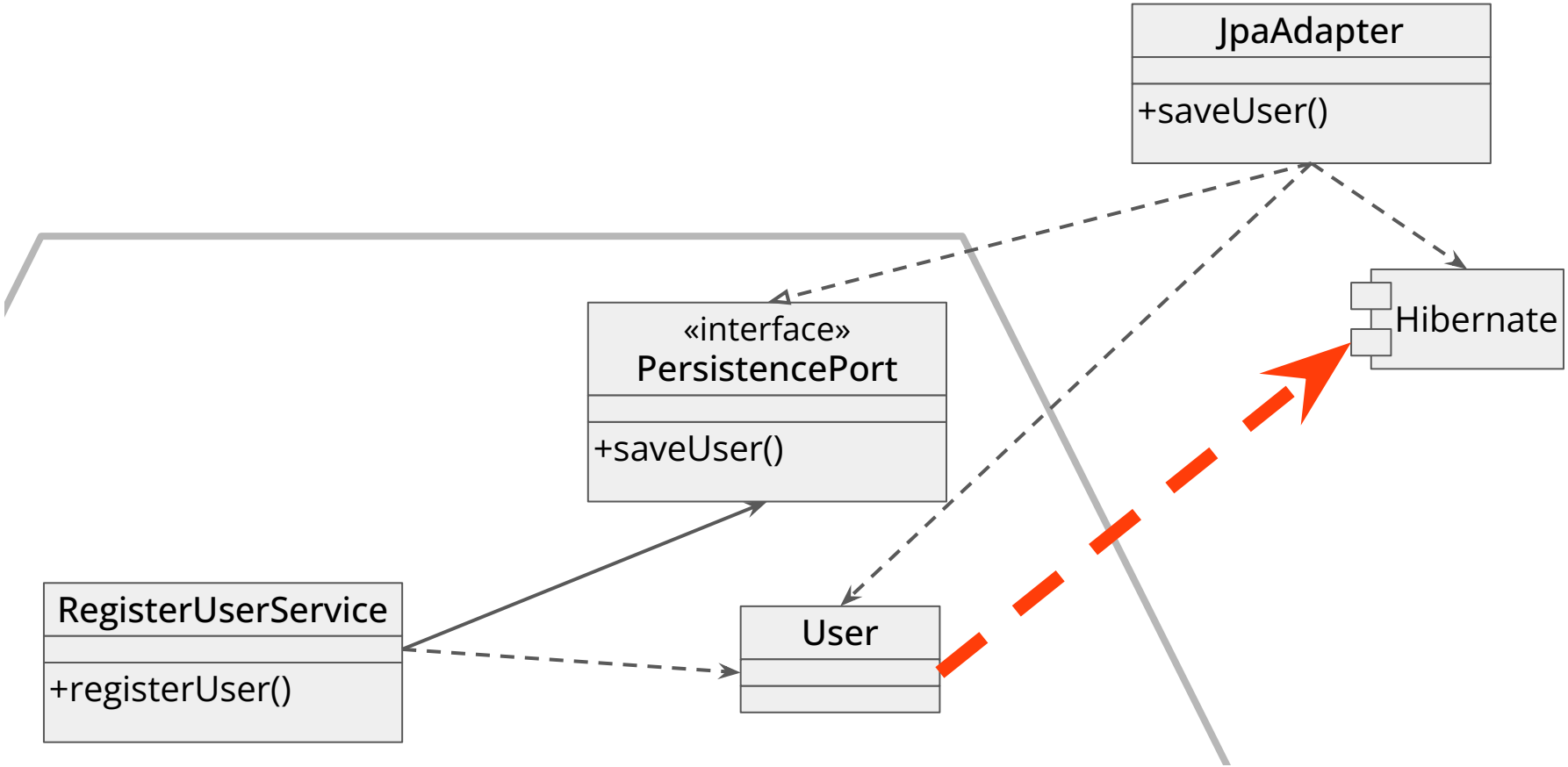


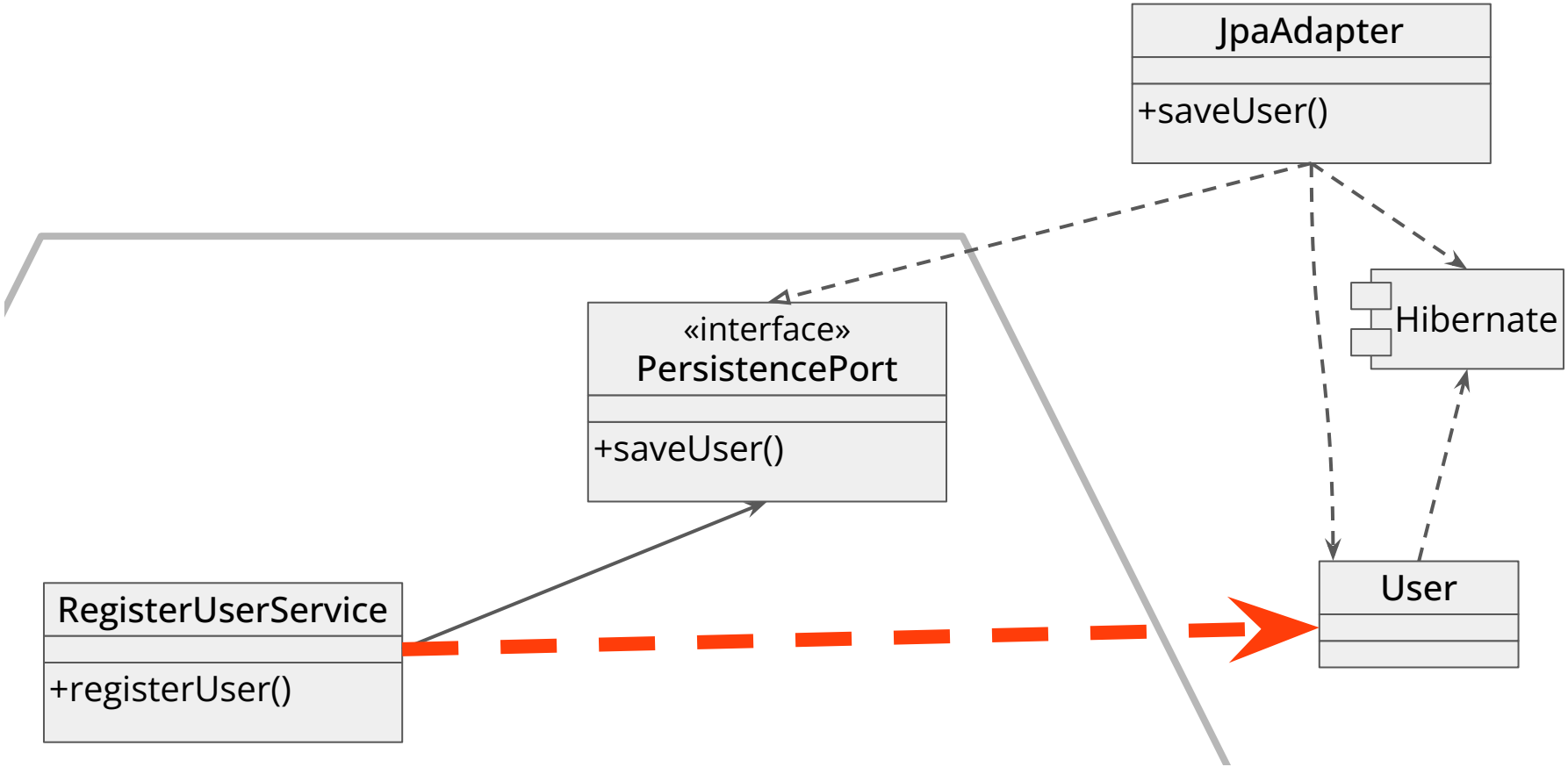


Mapping

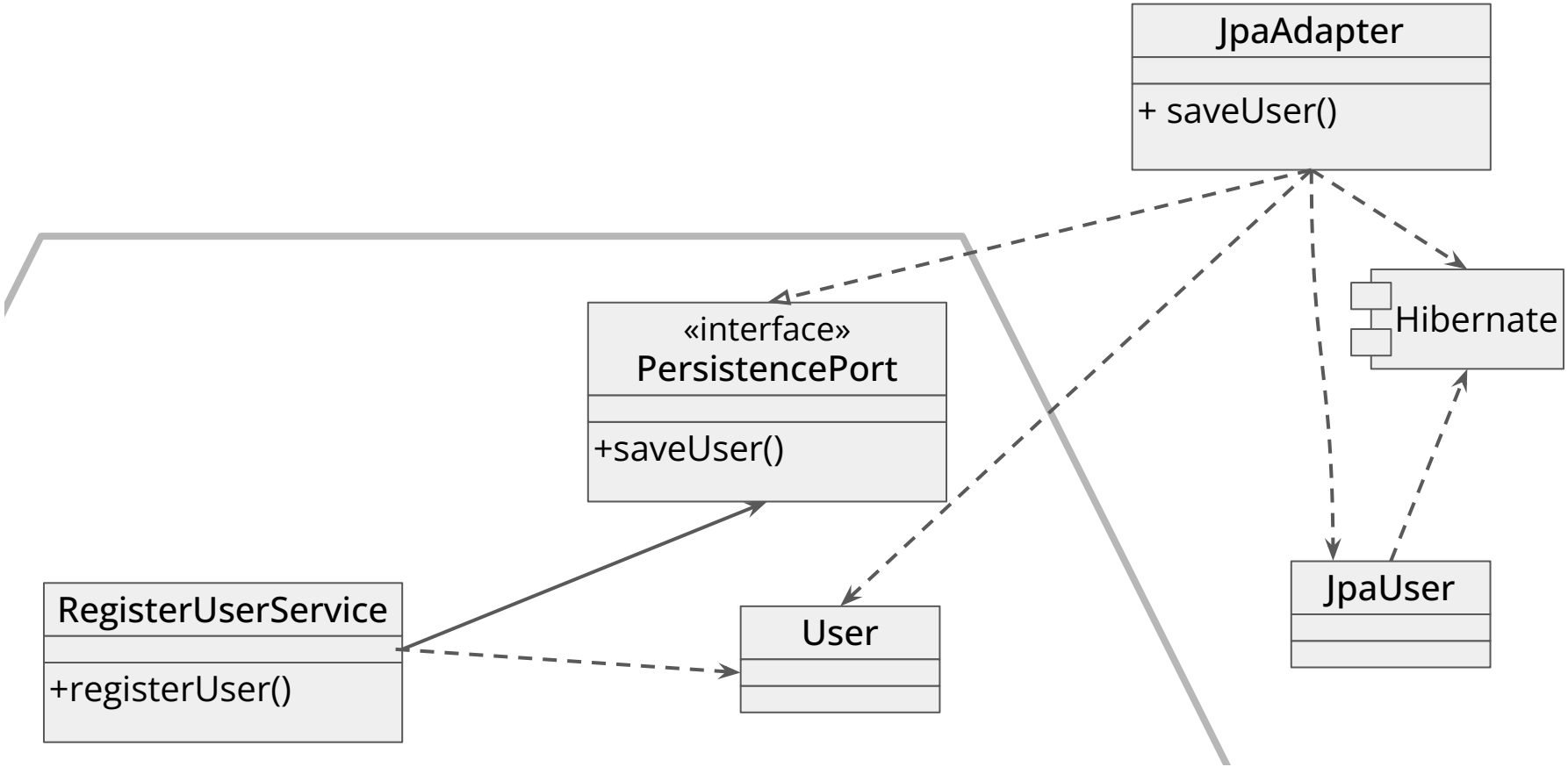
```java
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import java.time.LocalDateTime;

@Entity
public class User {

  @Id private Long id;

  @Column(nullable = false) private String name;
  @Column(nullable = false) private String email;
  @Column(nullable = false) private Instant registeredAt;

  // ...
}
```





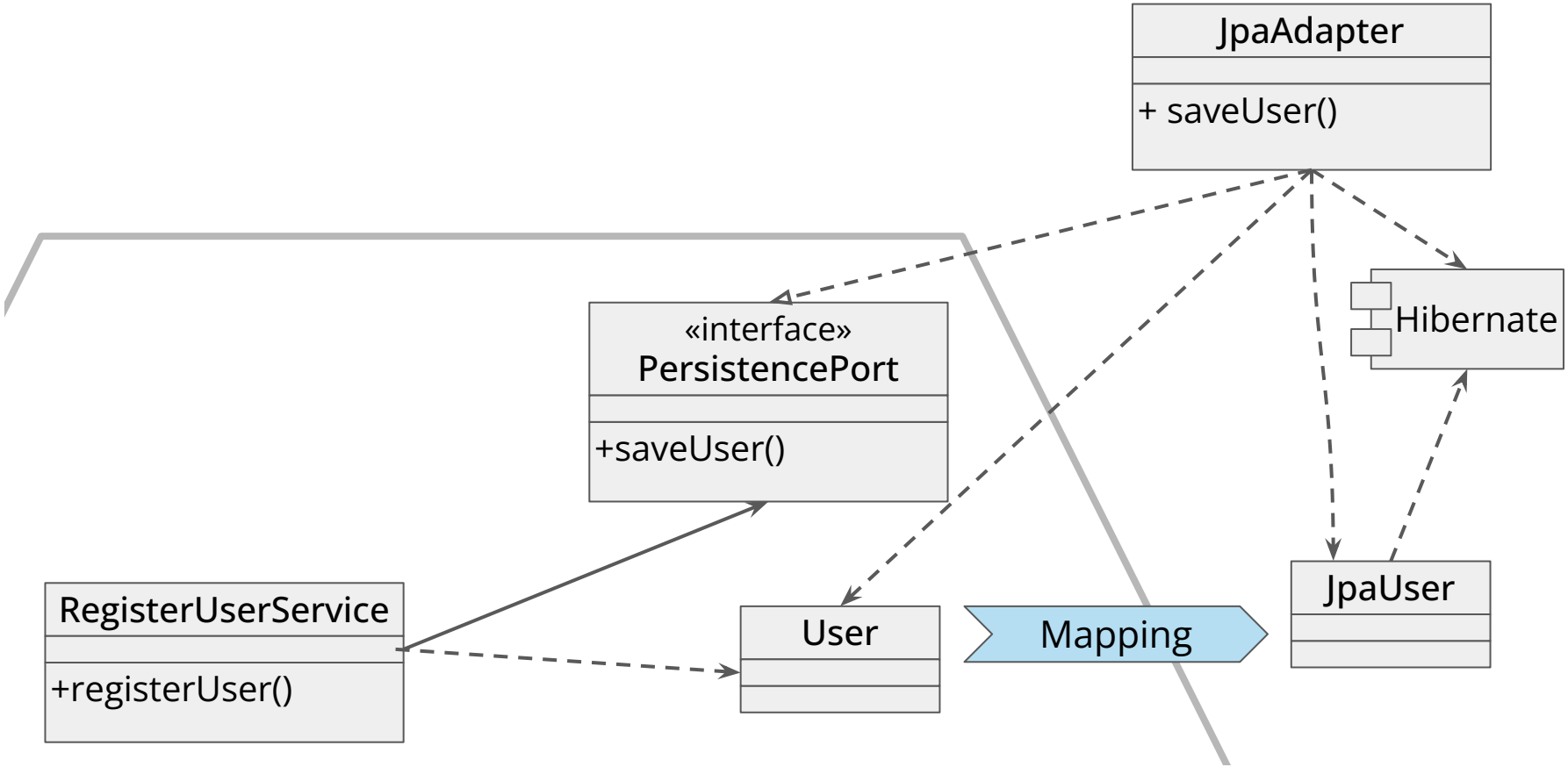
Two-Way Mapping

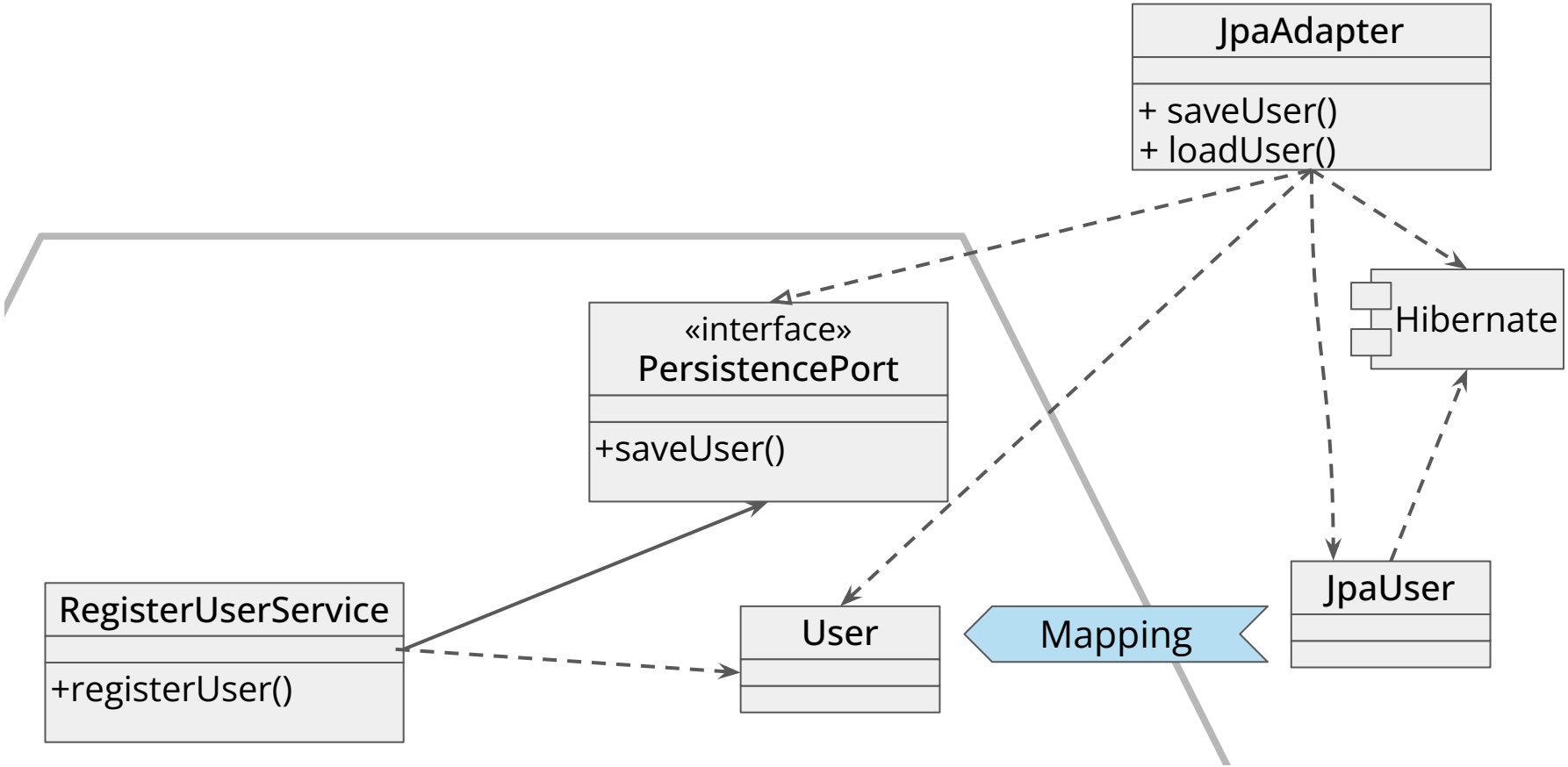


```java
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import java.time.Instant;

@Entity
public class JpaUser {

  @Id private Long id;

  @Column(nullable = false)
  private String name;

  @Column(nullable = false)
  private String email;

  @Column(nullable = false)
  private Instant registeredAt;

  // ...
}
```


```java
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import java.time.Instant;

@Entity
public class JpaUser {

  @Id private Long id;

  @Column(nullable = false)
  private String name;

  @Column(nullable = false)
  private String email;

  @Column(nullable = false)
  private Instant registeredAt;

  // ...
}
```

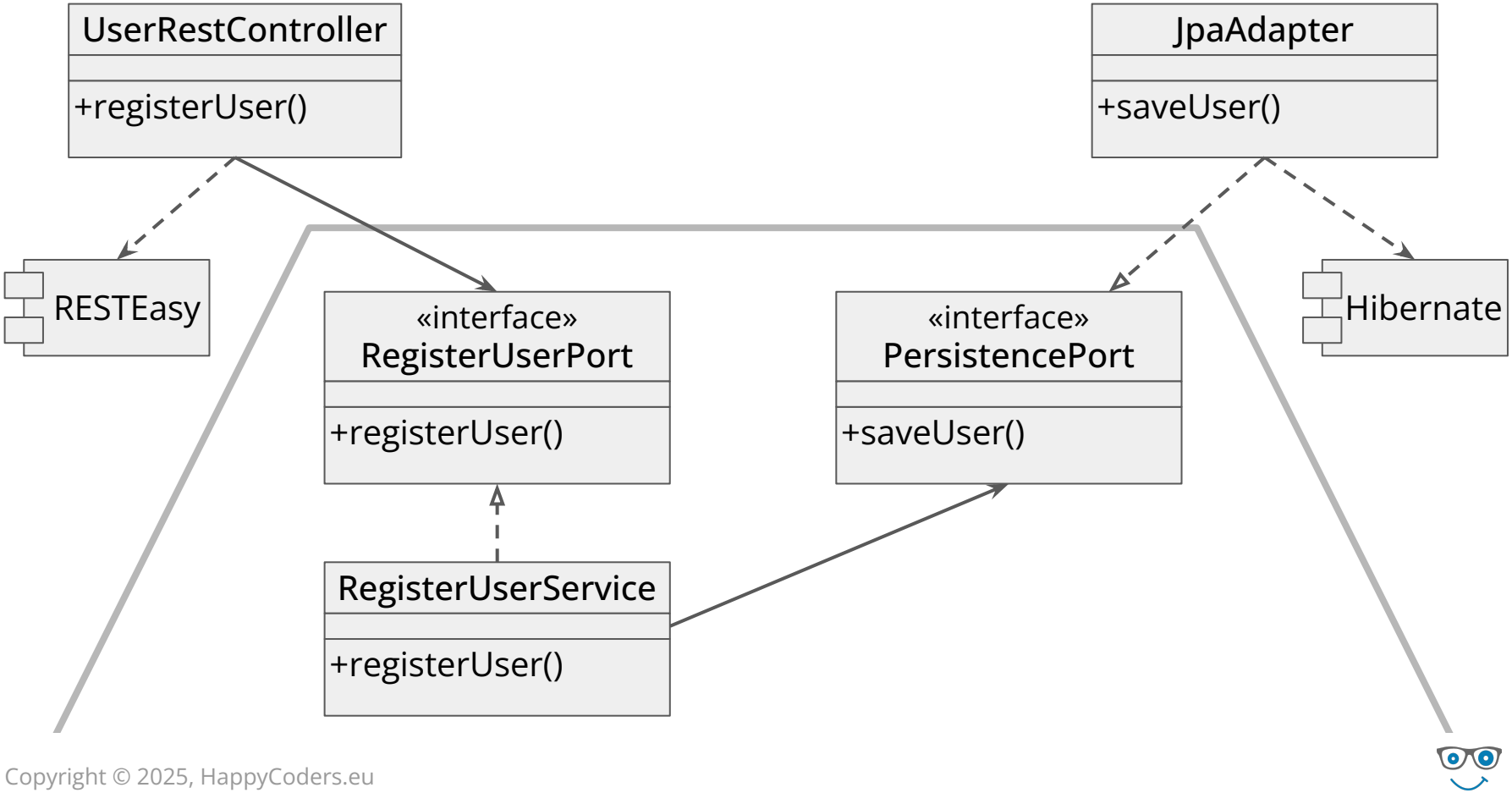
```java
import java.time.Instant;

public class User {

  private Long id;
  private String name;
  private String email;
  private Instant registeredAt;

  public User(String name, String email) {
    verifyName(name);
    verifyEmail(email);

    this.name = name;
    this.email = email;
    this.registeredAt = Instant.now();
  }

  // ...
}
```

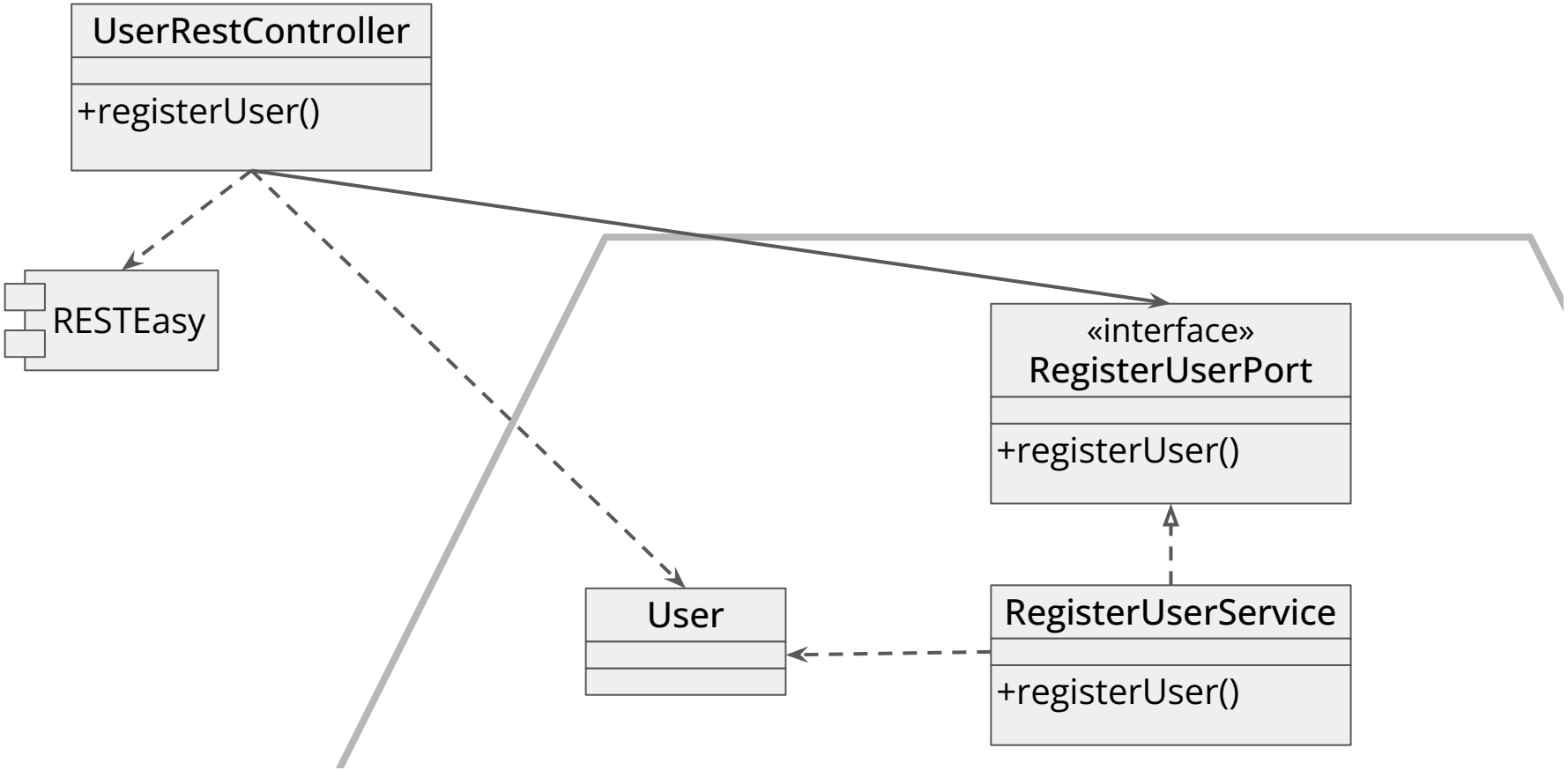





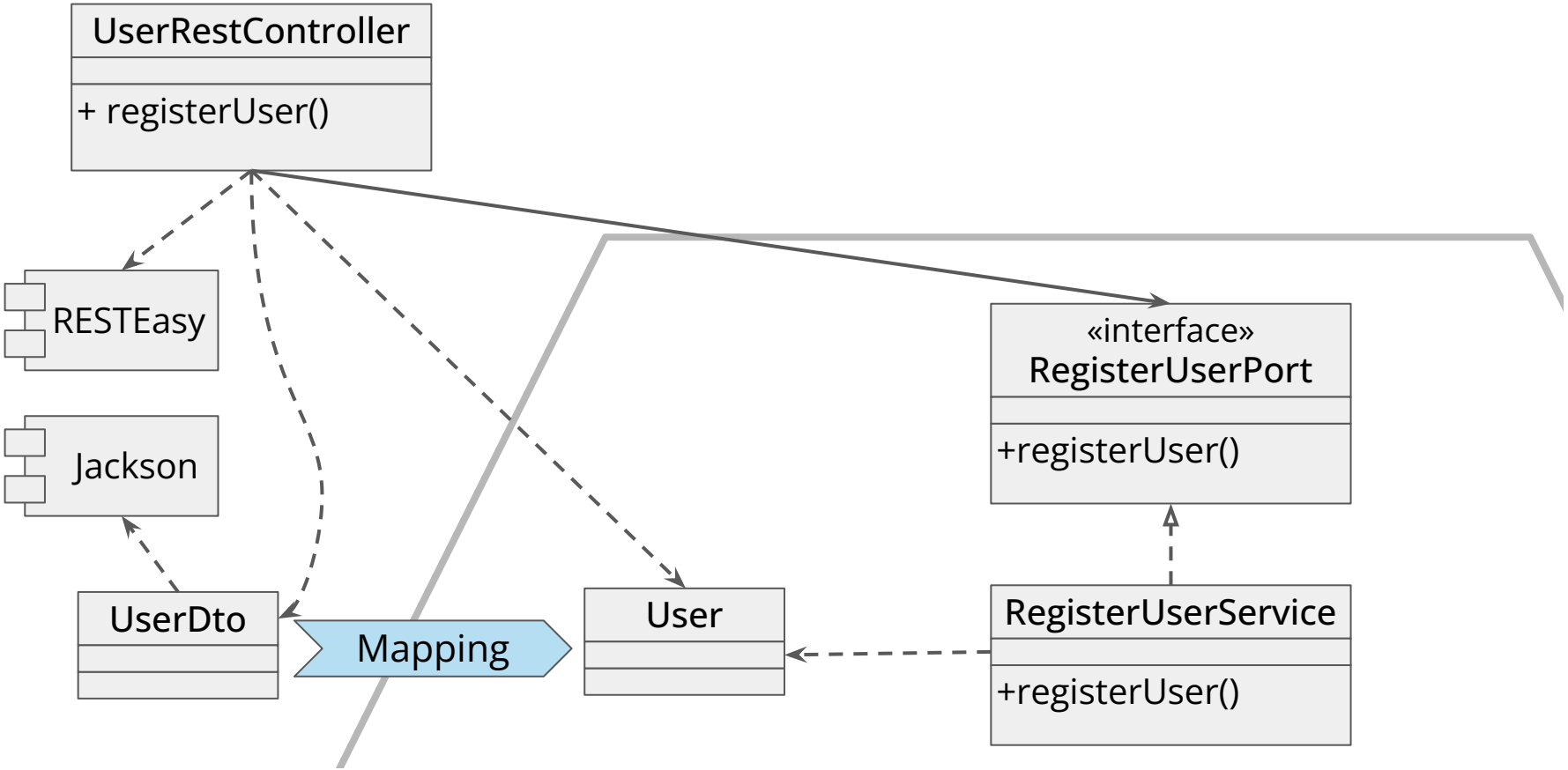
```java
public class JpaAdapter implements PersistencePort {

  @Override
  @Transactional
  public void saveUser(User user) {
    JpaUser jpaUser = mapper.toJpaUser(user);
    repository.save(jpaUser);
  }

  @Override
  @Transactional
  public Optional<User> loadUser(long userId) {
    Optional<JpaUser> jpaUser = repository.findById(userId);
    return jpaUser.map(mapper::toUser);
  }

}
```

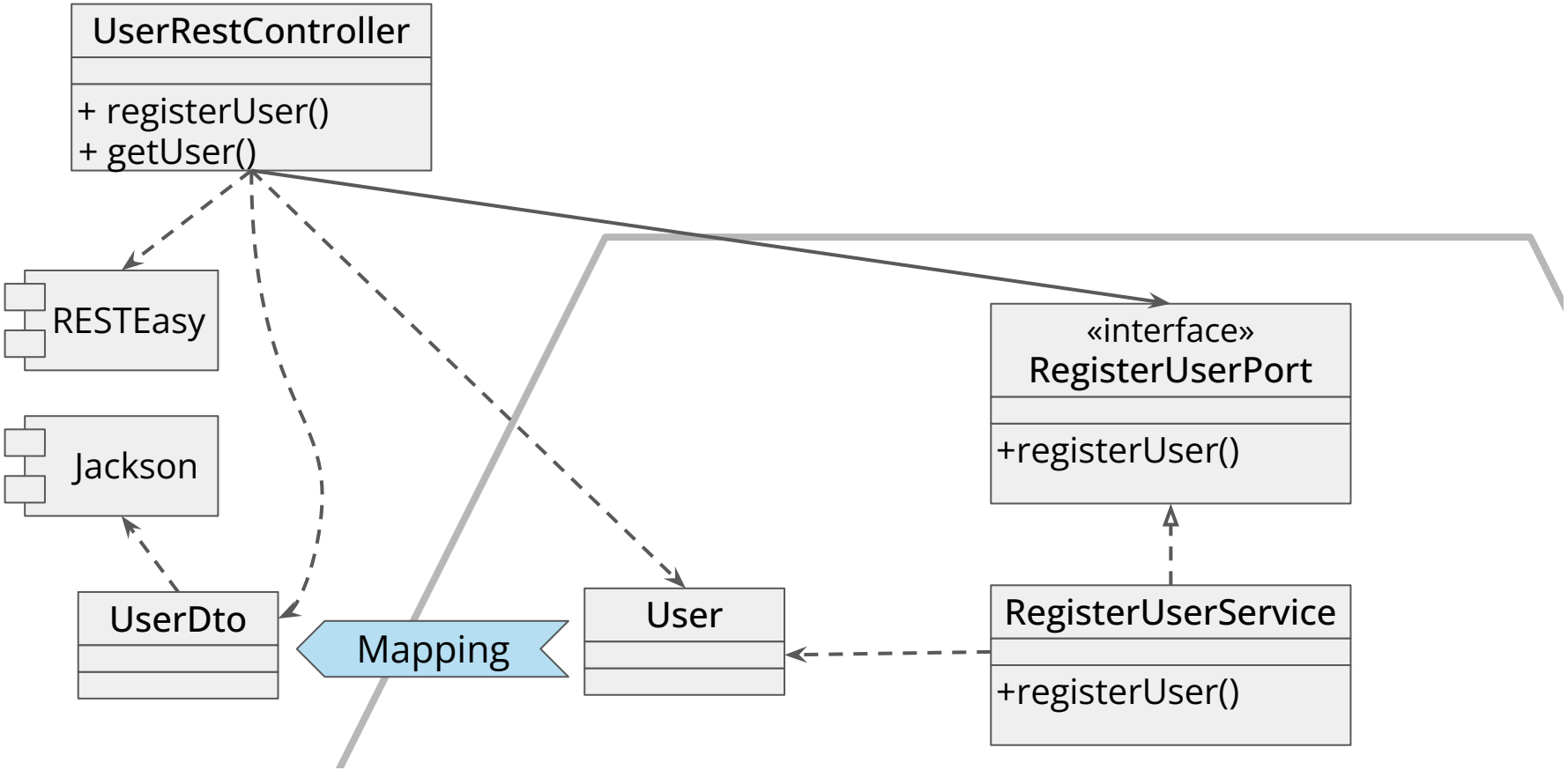

```java
public class JpaAdapter implements PersistencePort {

  @Override
  @Transactional
  public void saveUser(User user) {
    JpaUser jpaUser = mapper.toJpaUser(user);
    repository.save(jpaUser);
  }

  @Override
  @Transactional
  public Optional<User> loadUser(long userId) {
    Optional<JpaUser> jpaUser = repository.findById(userId);
    return jpaUser.map(mapper::toUser);
  }

}
```



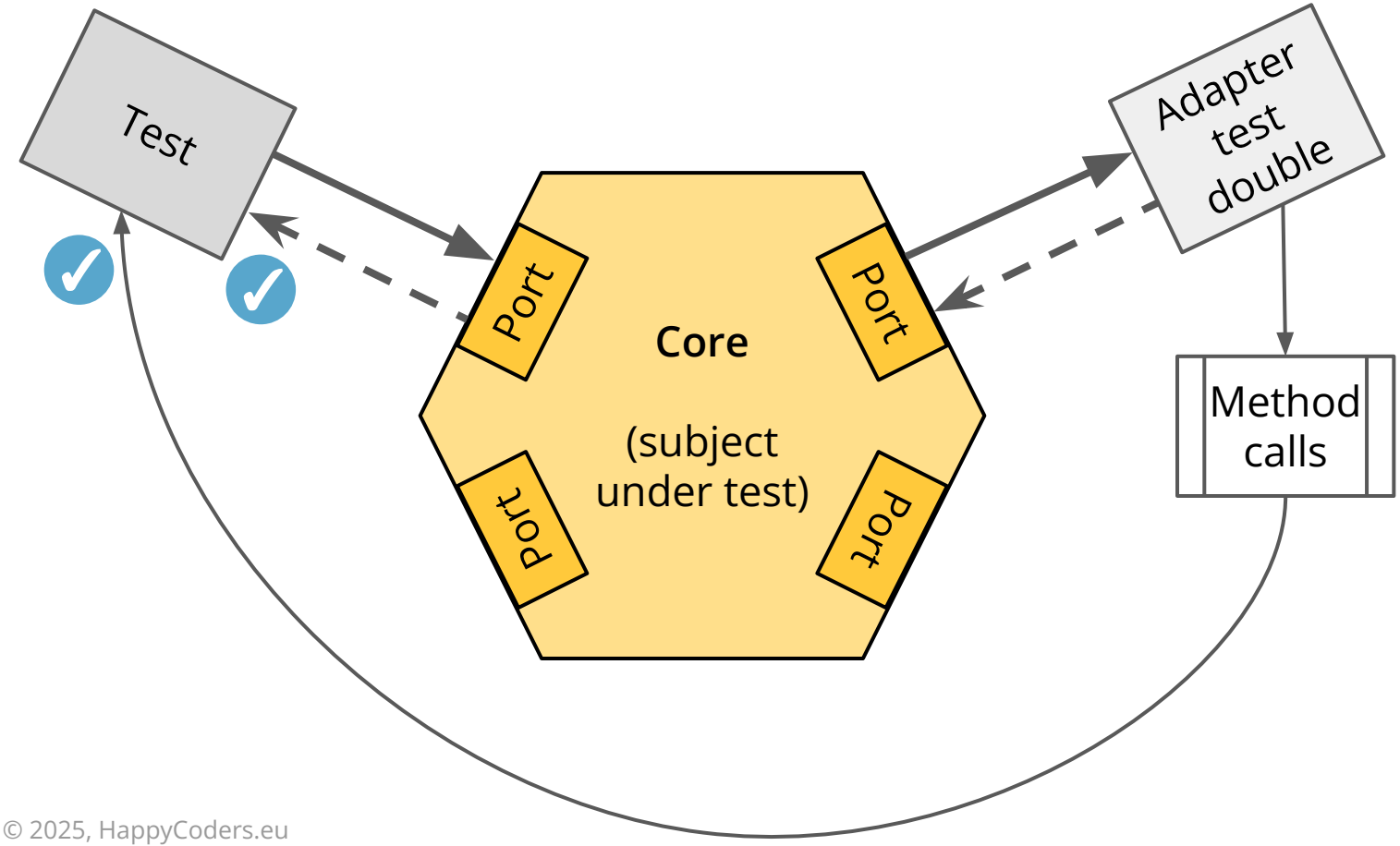


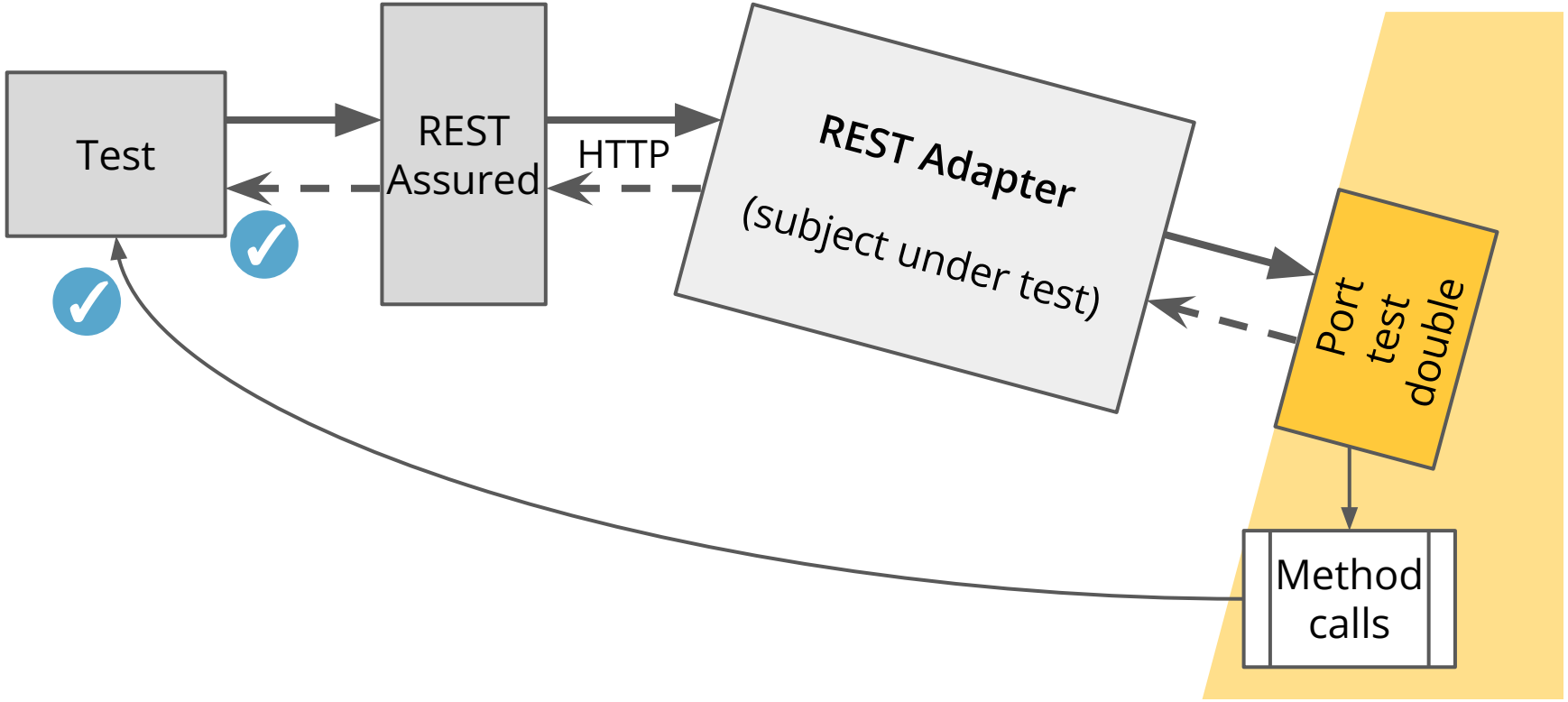
```java
import com.fasterxml.jackson.annotation.JsonFormat;
import java.time.Instant;

public class UserDto {

  private Long id;
  private String name;
  private String email;

  @JsonFormat(pattern = "yyyy-MM-dd'T'HH:mm:ss'Z'")
  private Instant registeredAt;

  // ...
}
```

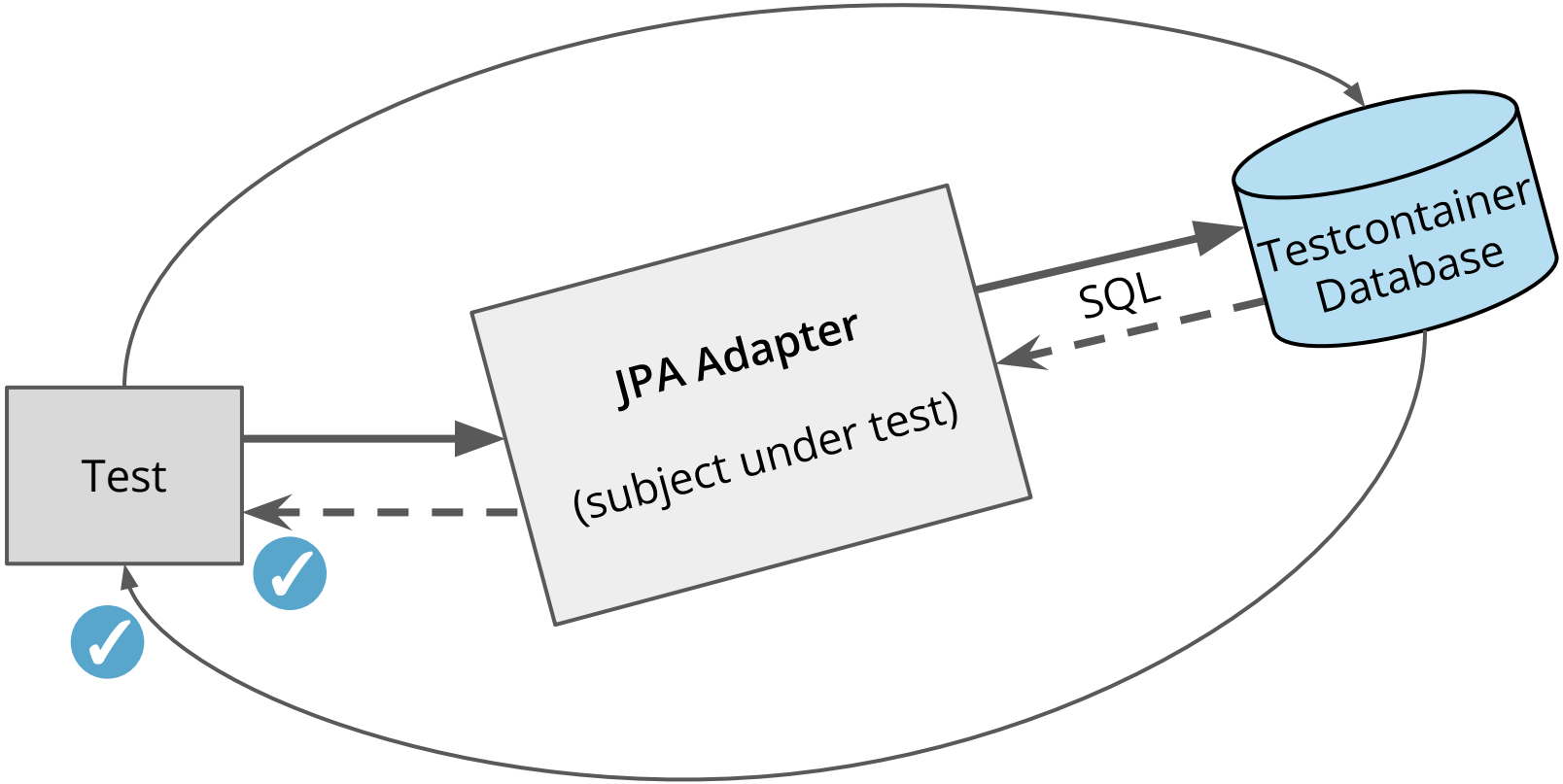



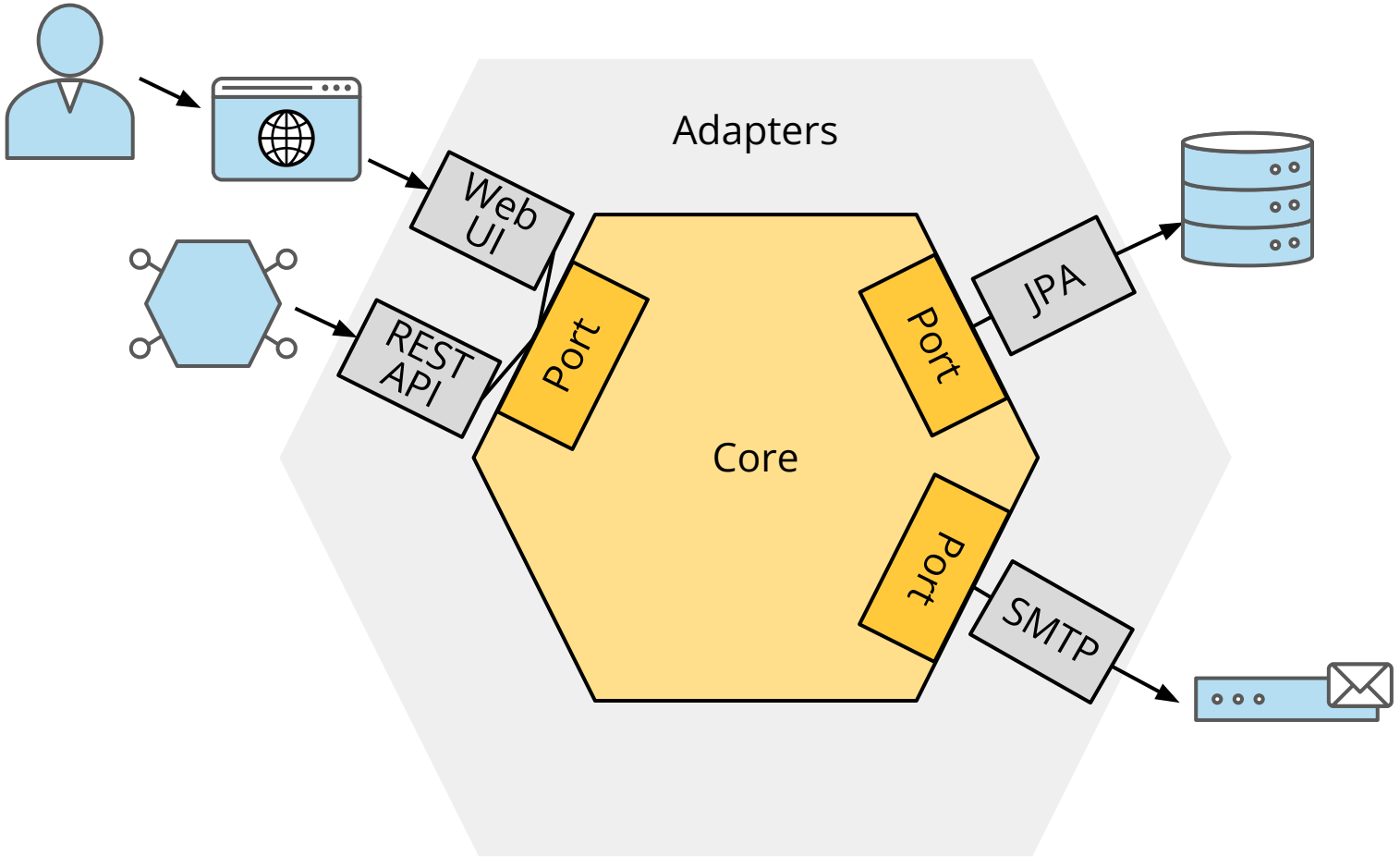


Testability



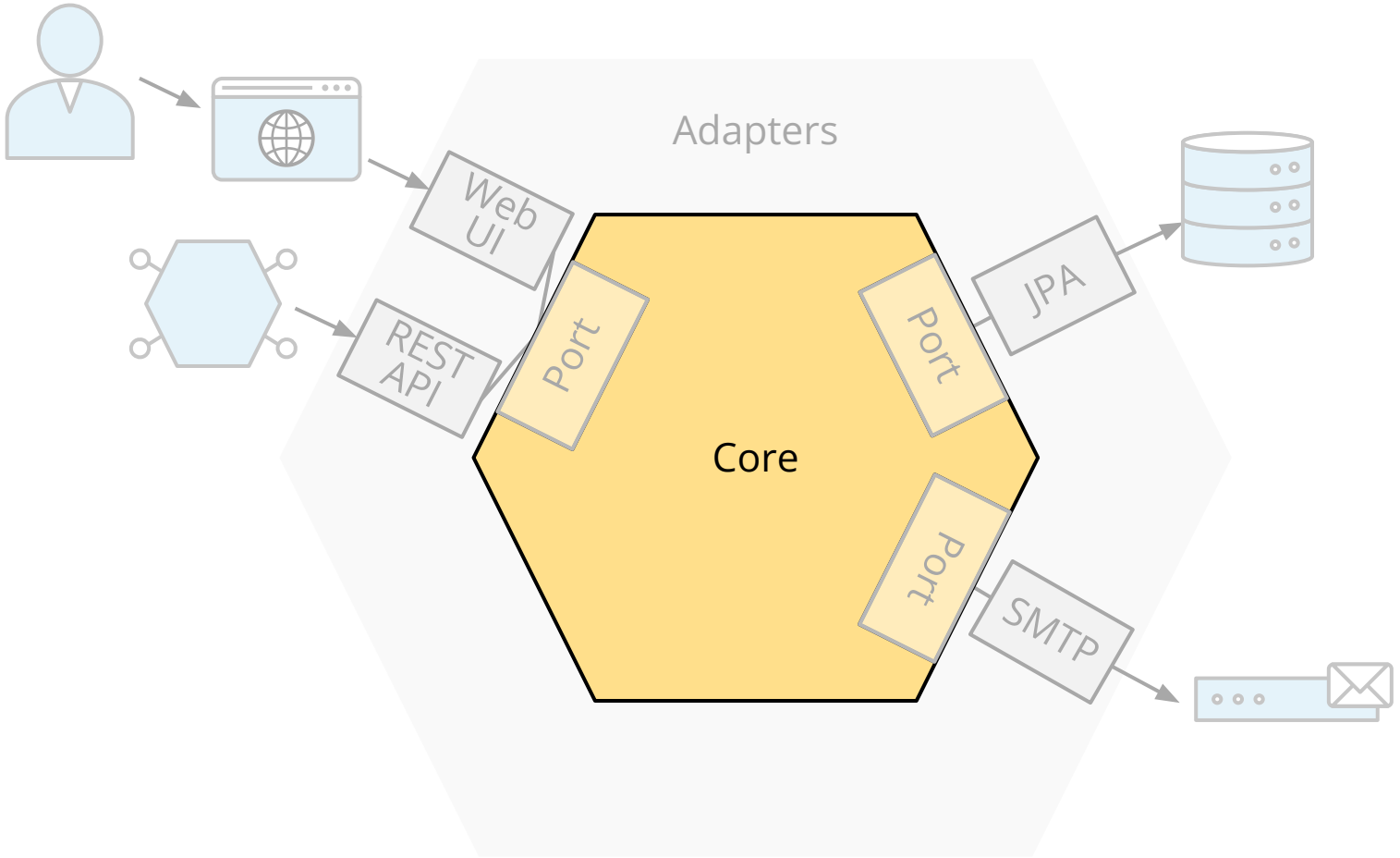


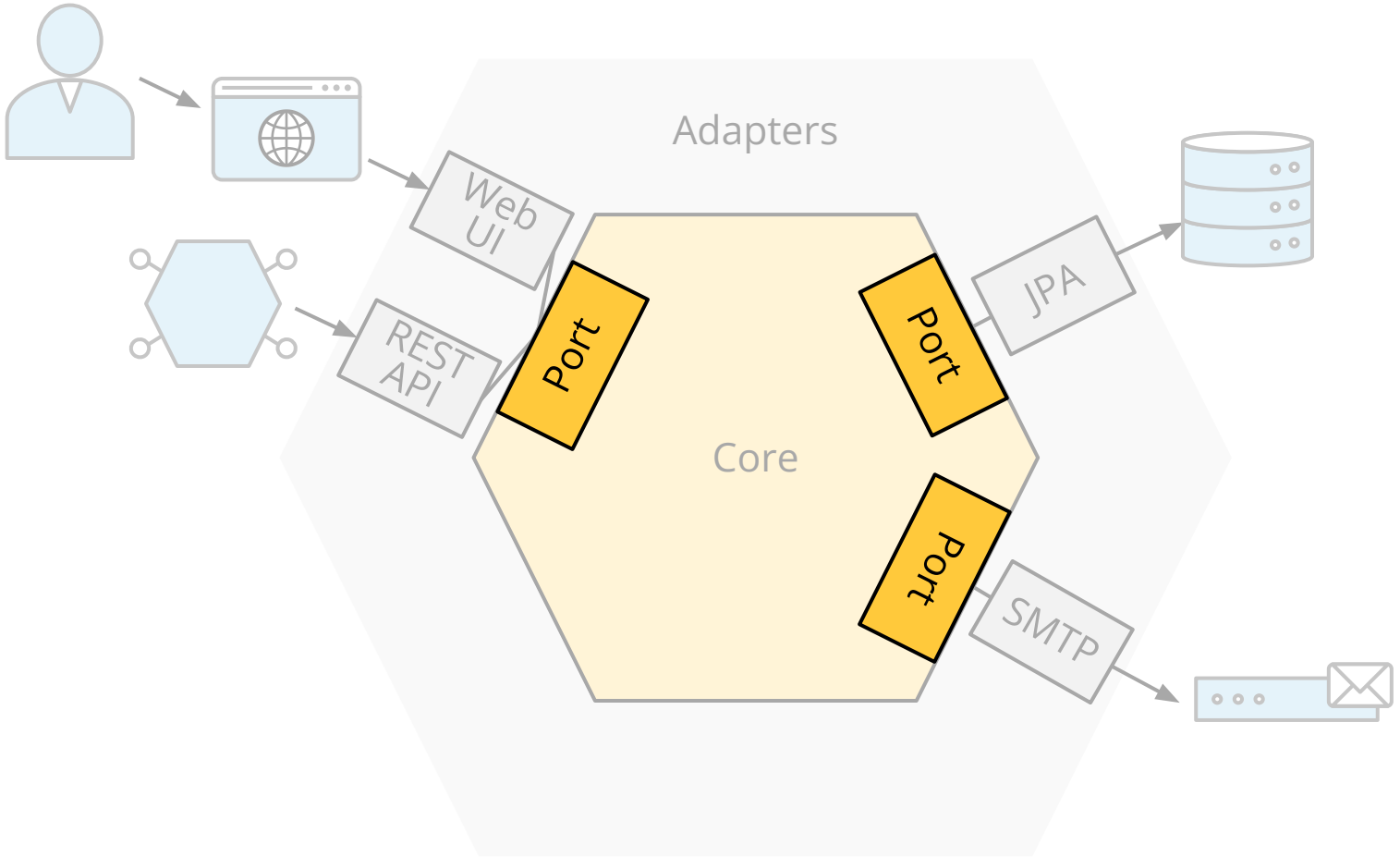


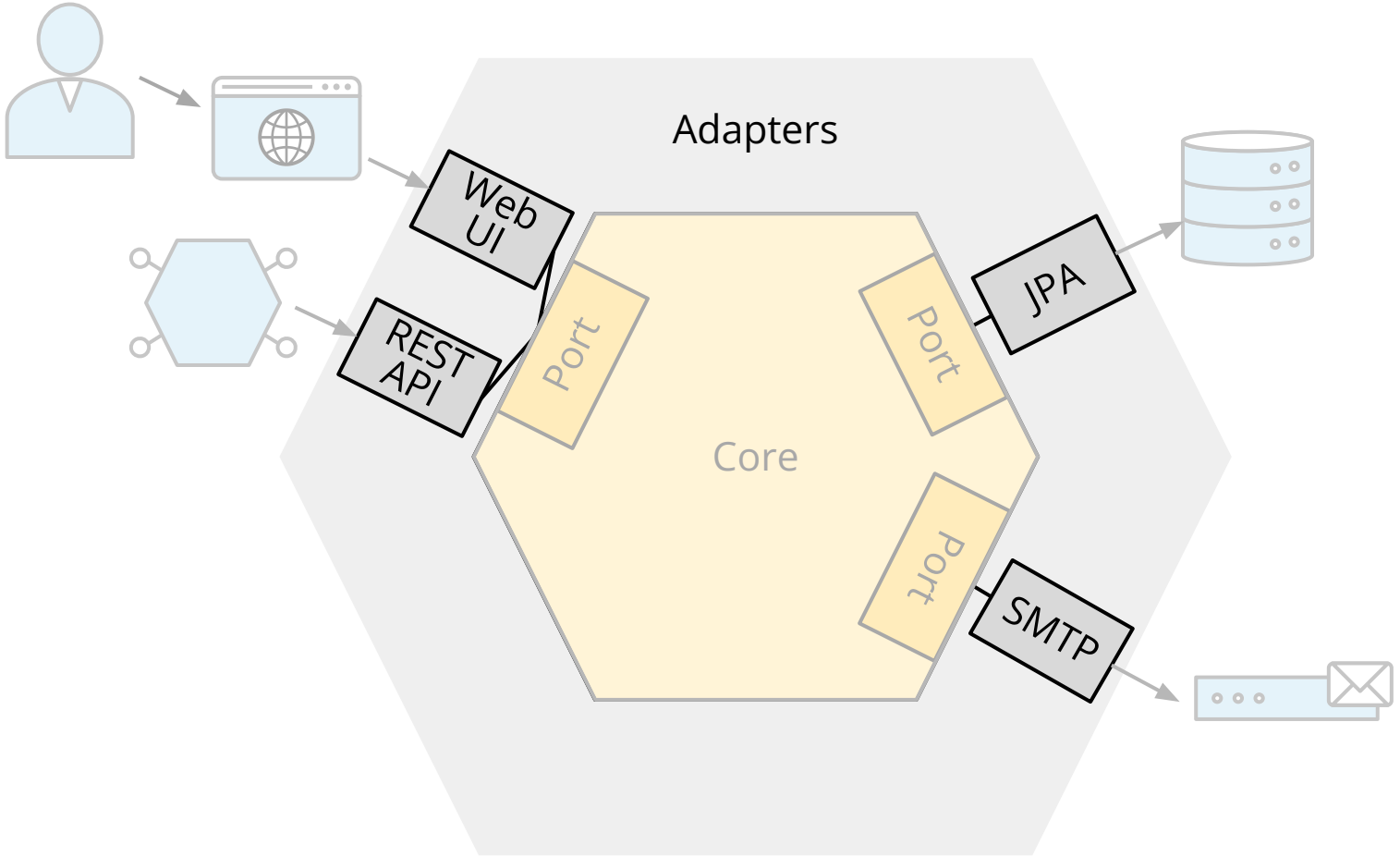


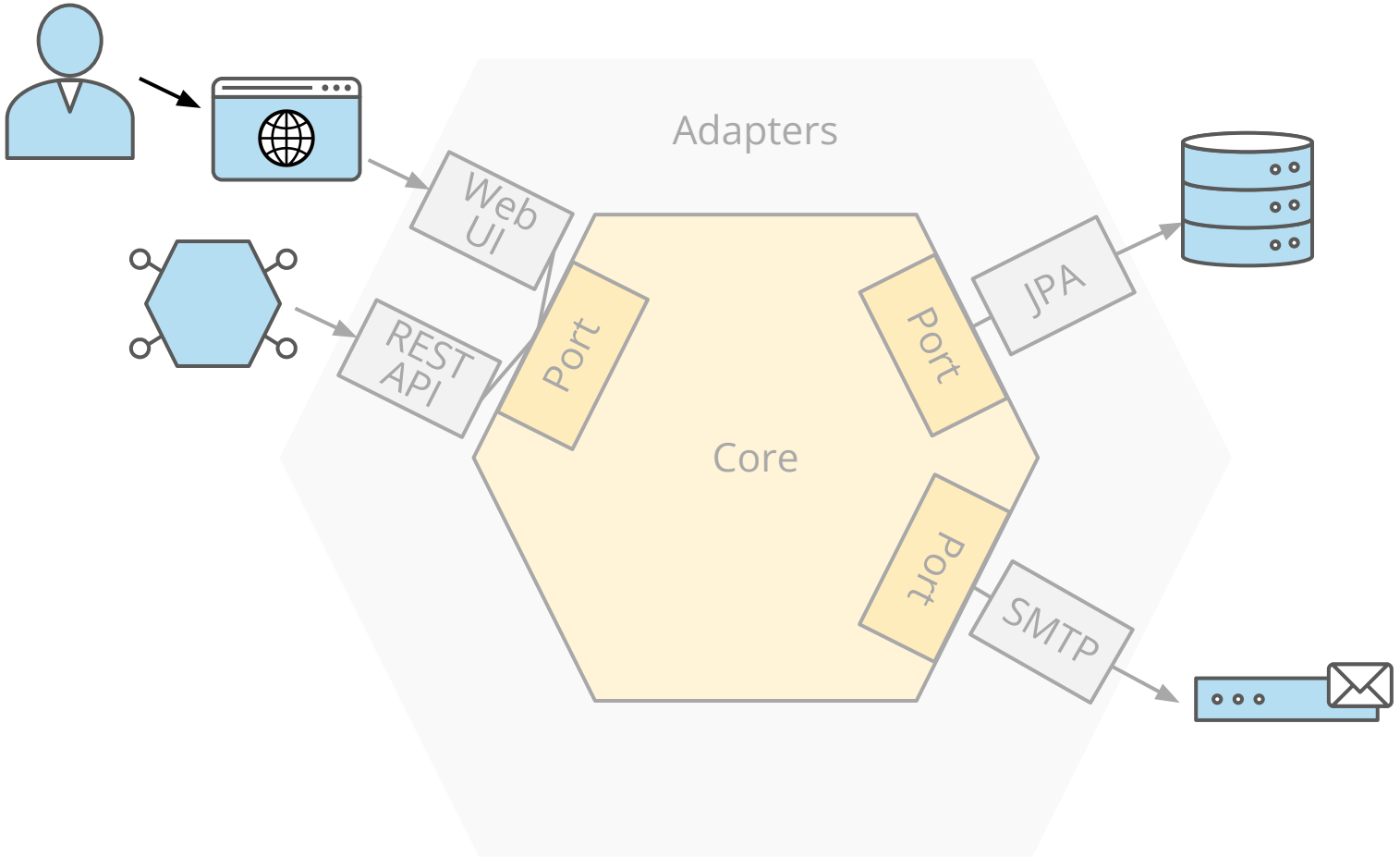
Hexagonale Architektur
vs.
„Ports & Adapters“

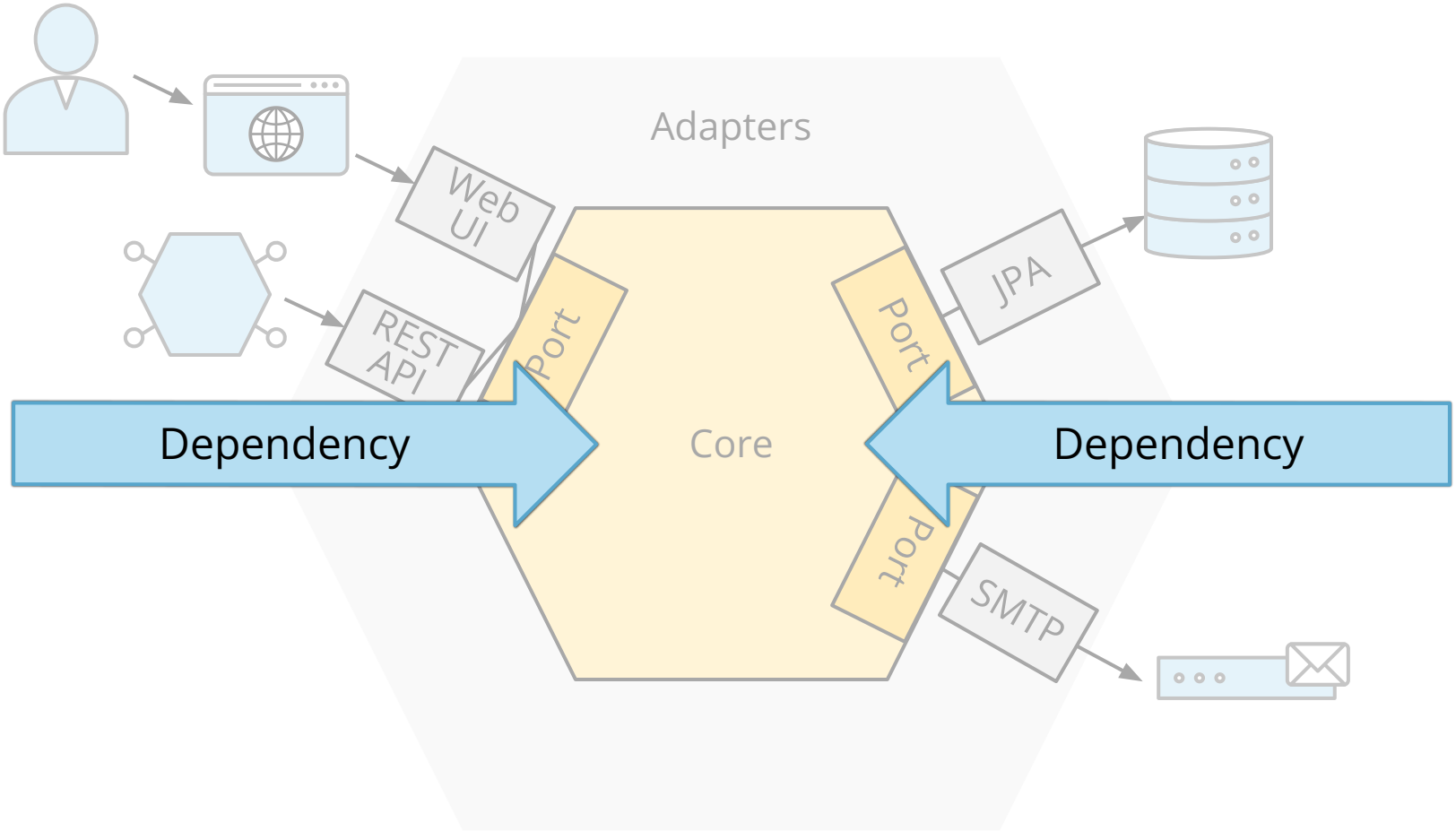
Hexagonale Architektur
=
„Ports & Adapters“







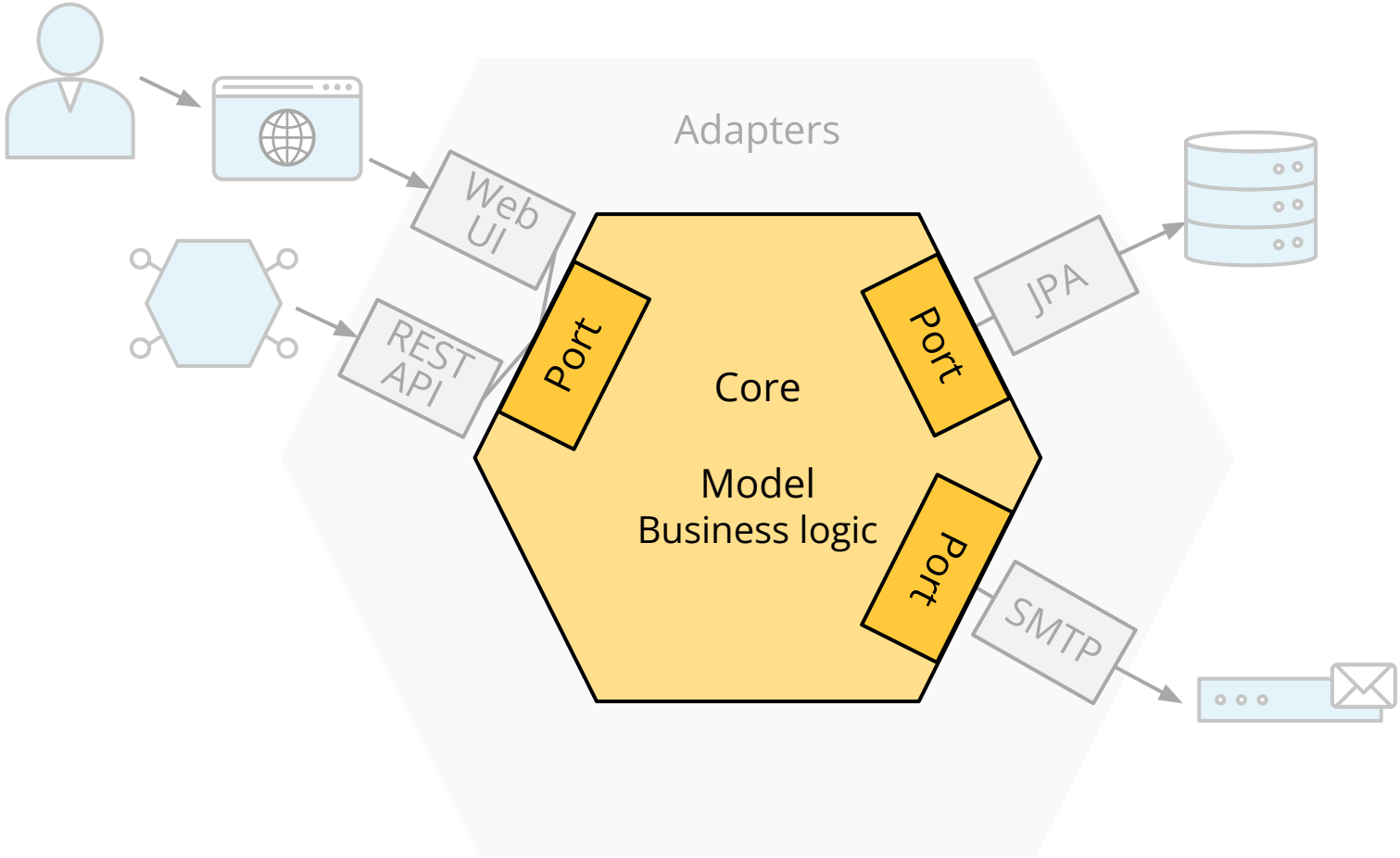


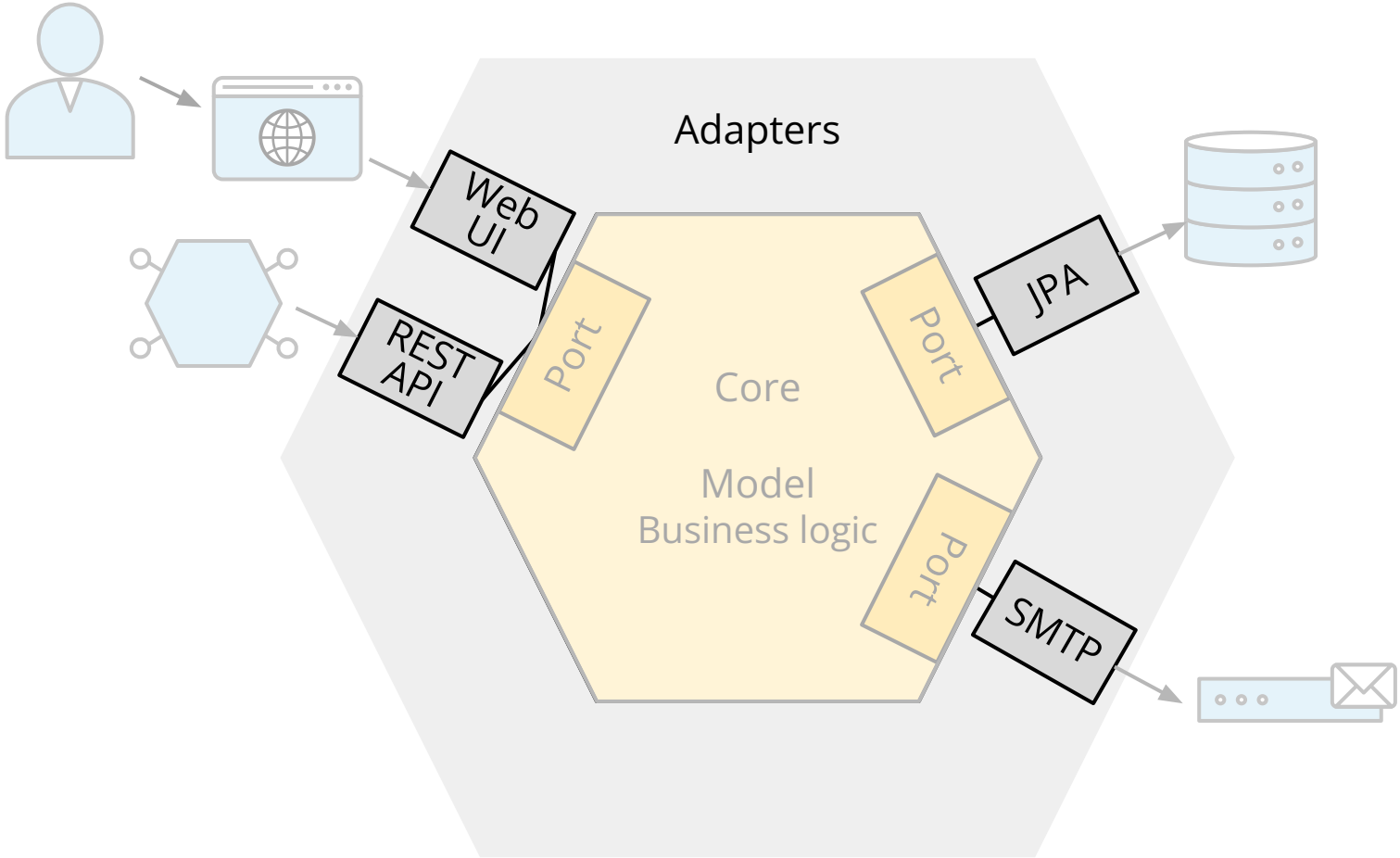


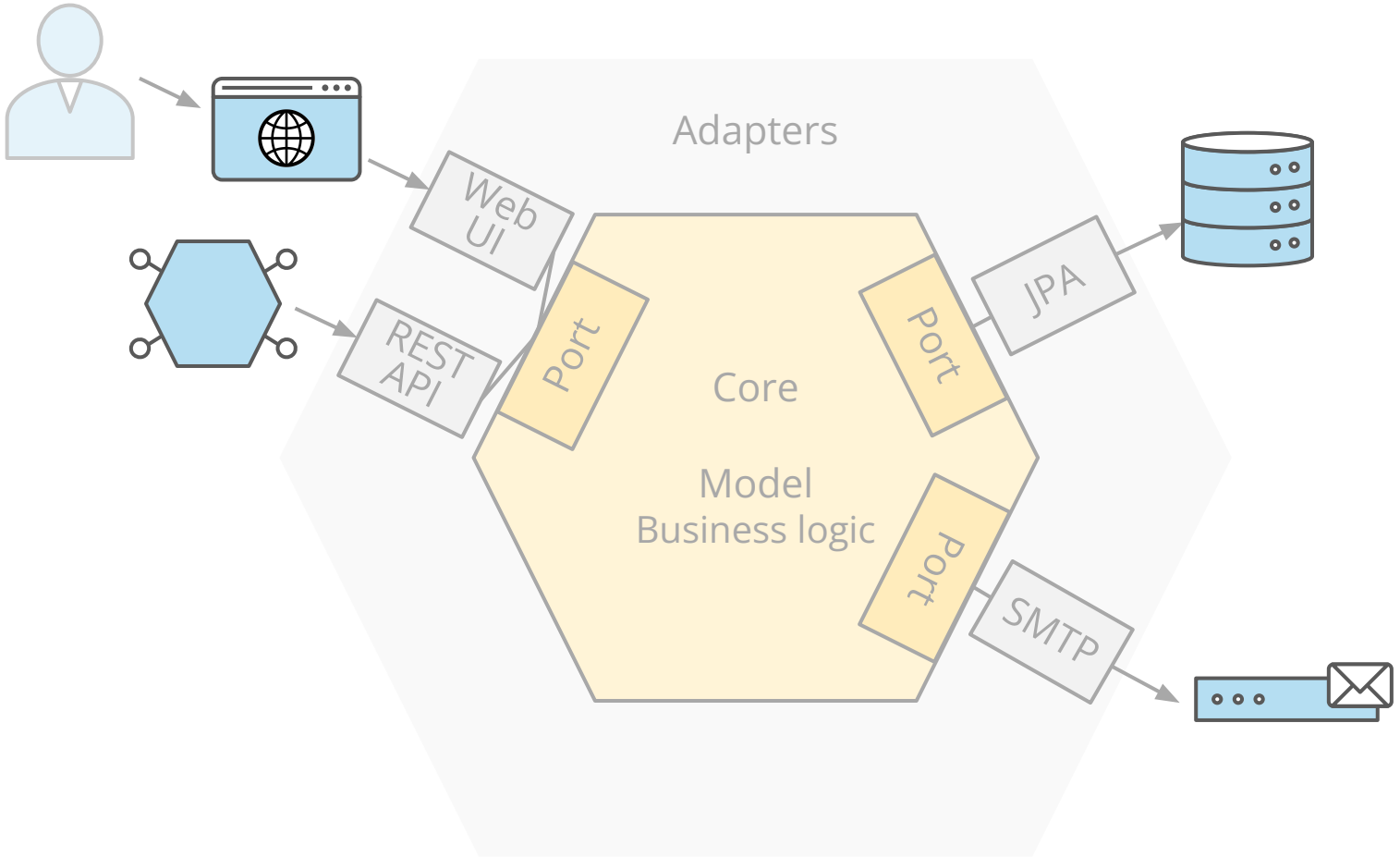
What Is the Goal of Software Architecture?

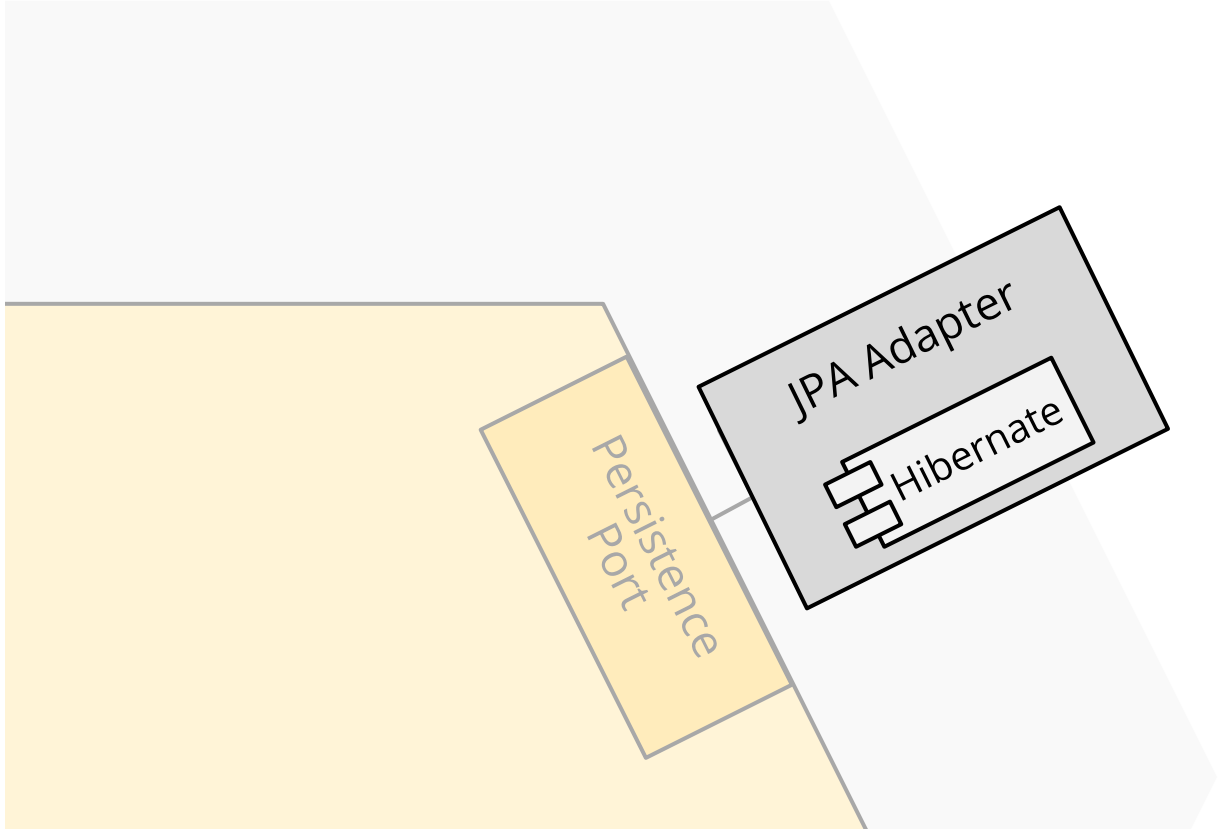
A good architecture makes it possible to change software during its lifetime with as little effort as possible.

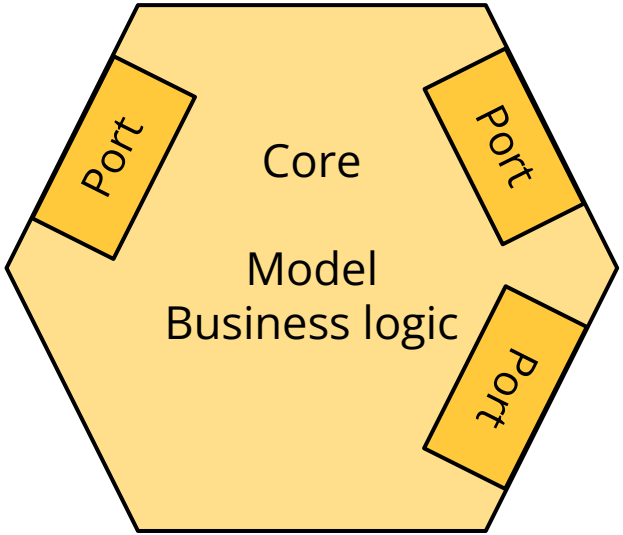
Goal #1: Changeability



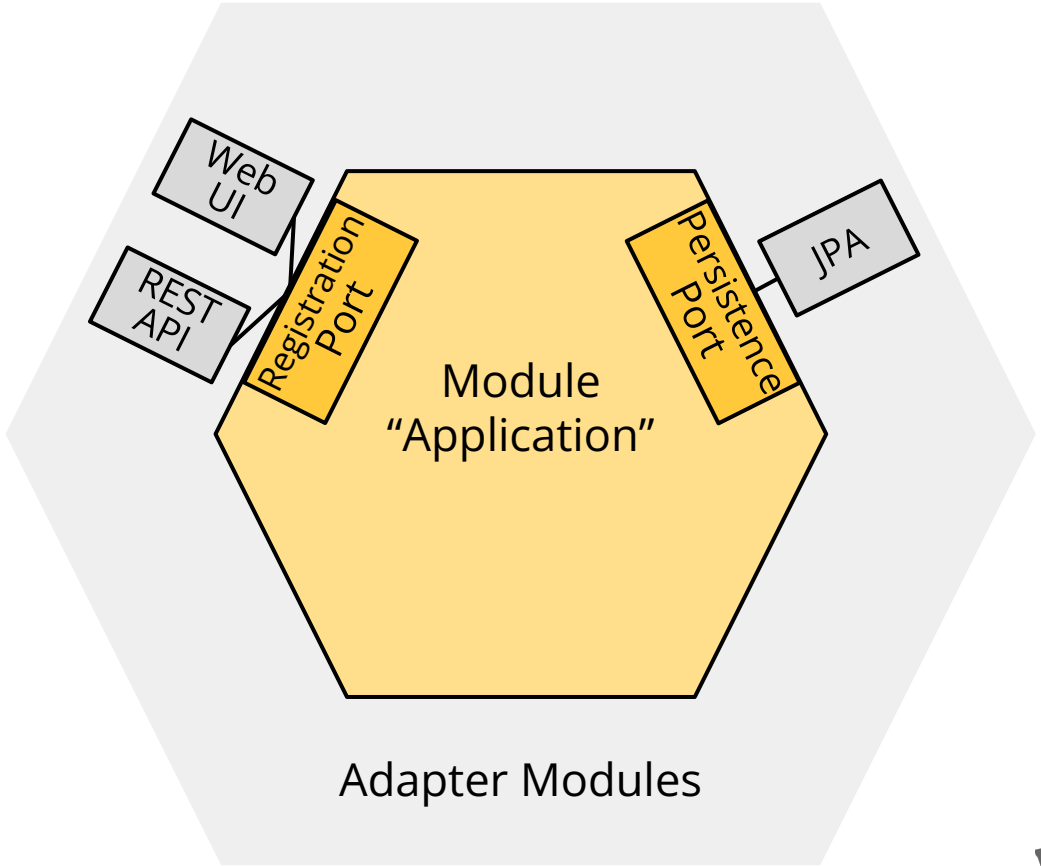


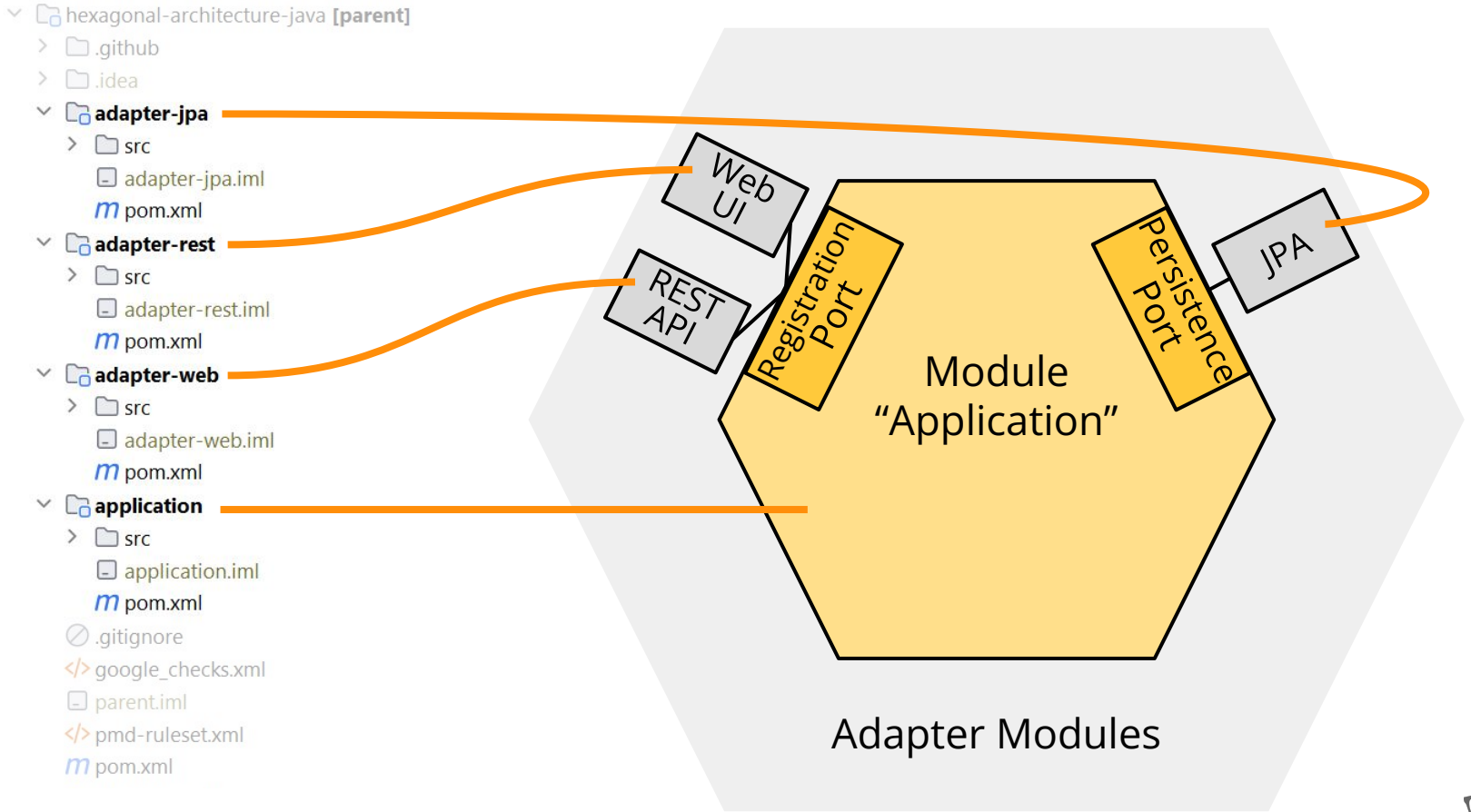


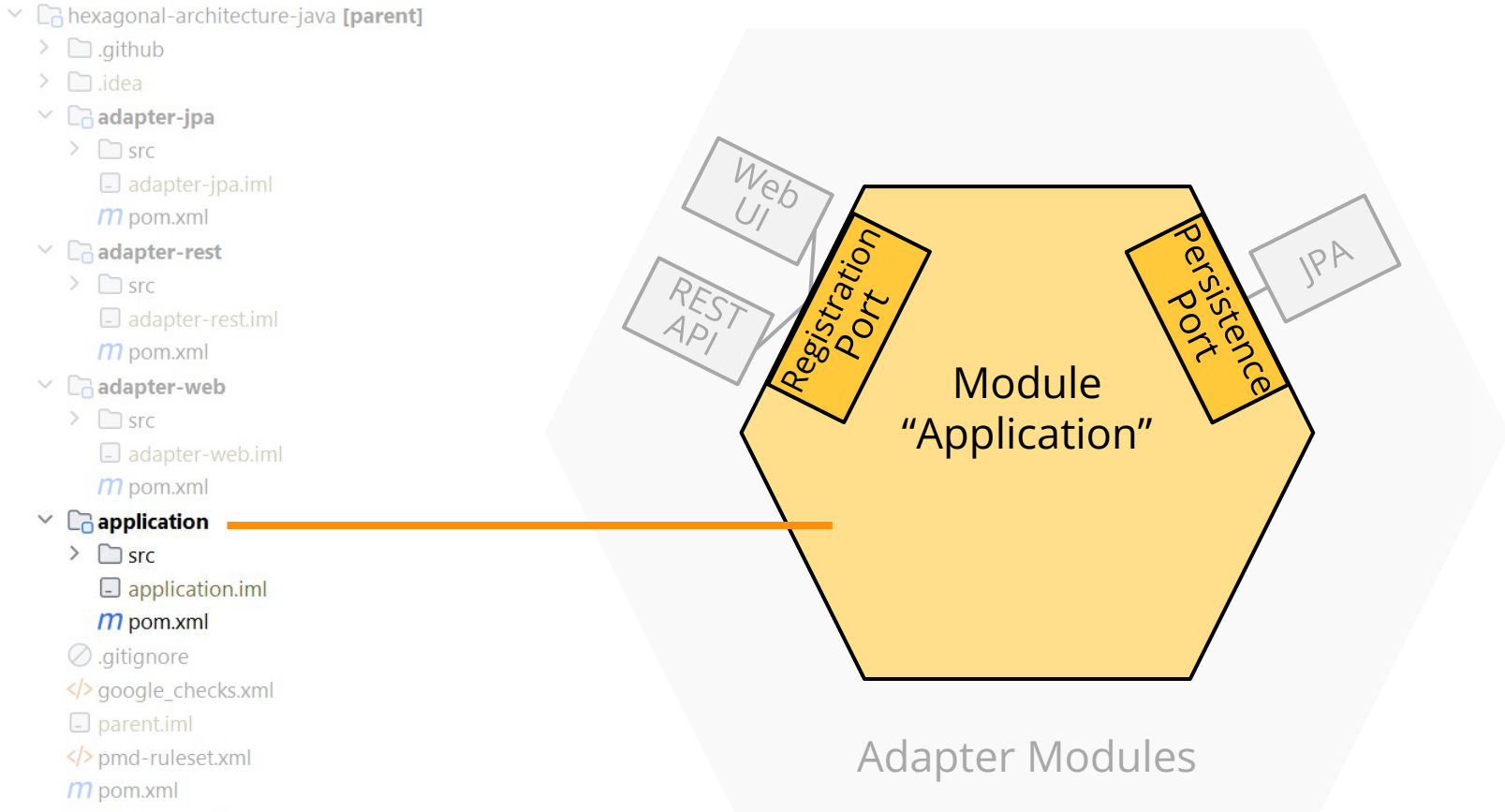


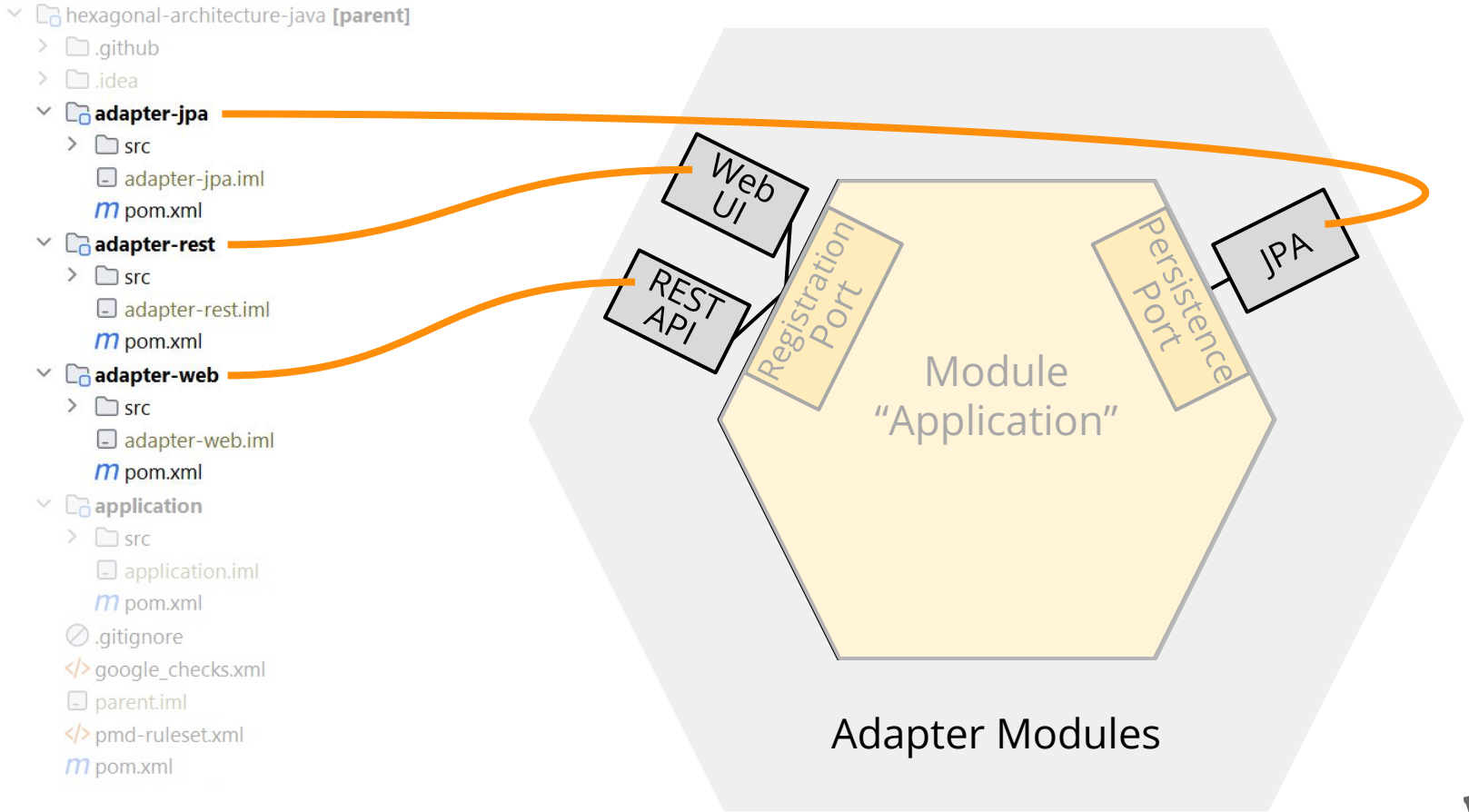


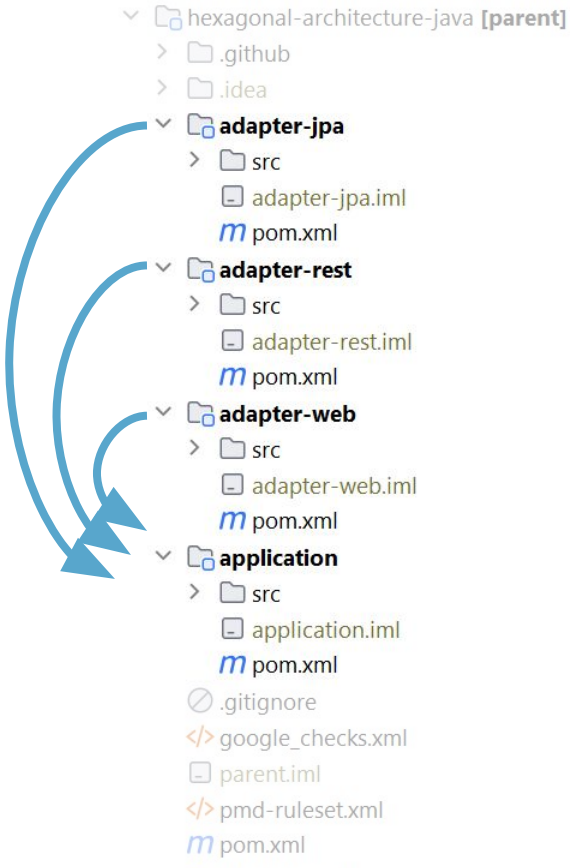
Goal #2: Loose Coupling

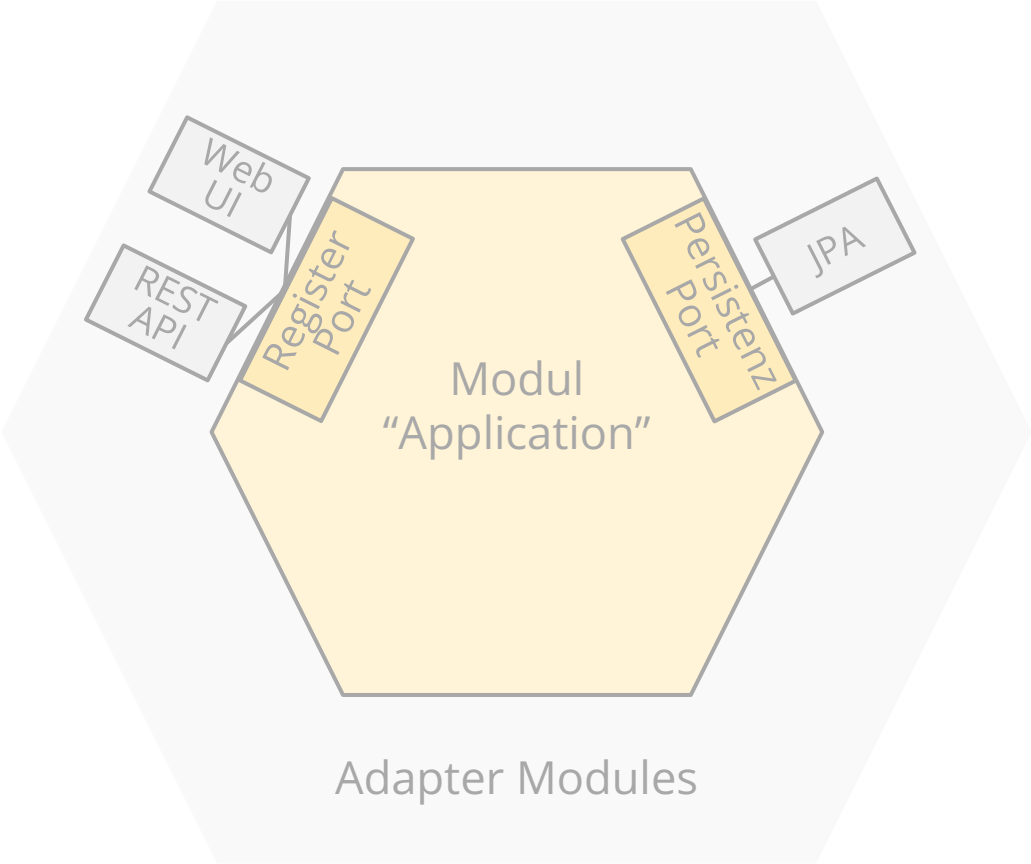


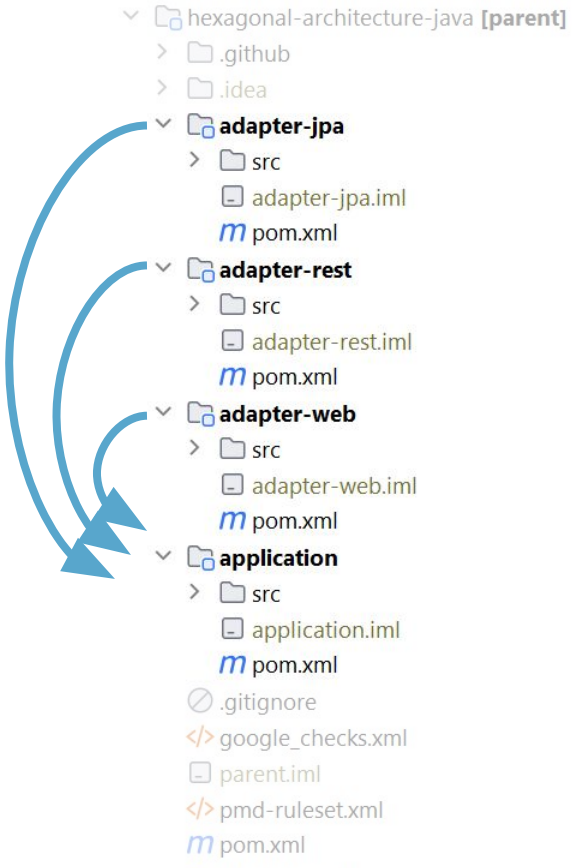


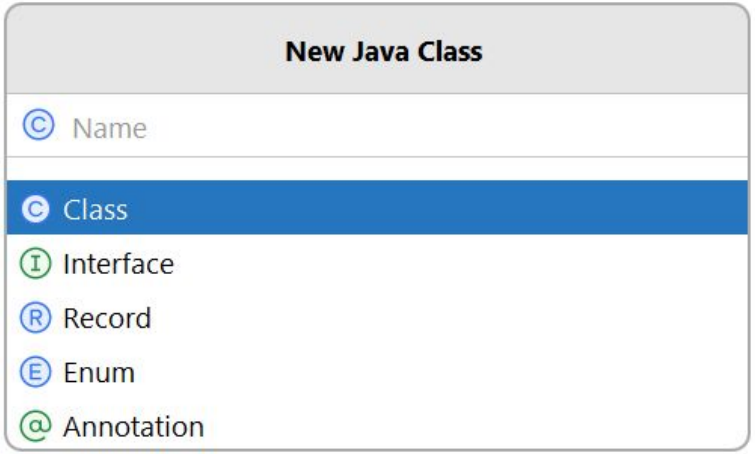




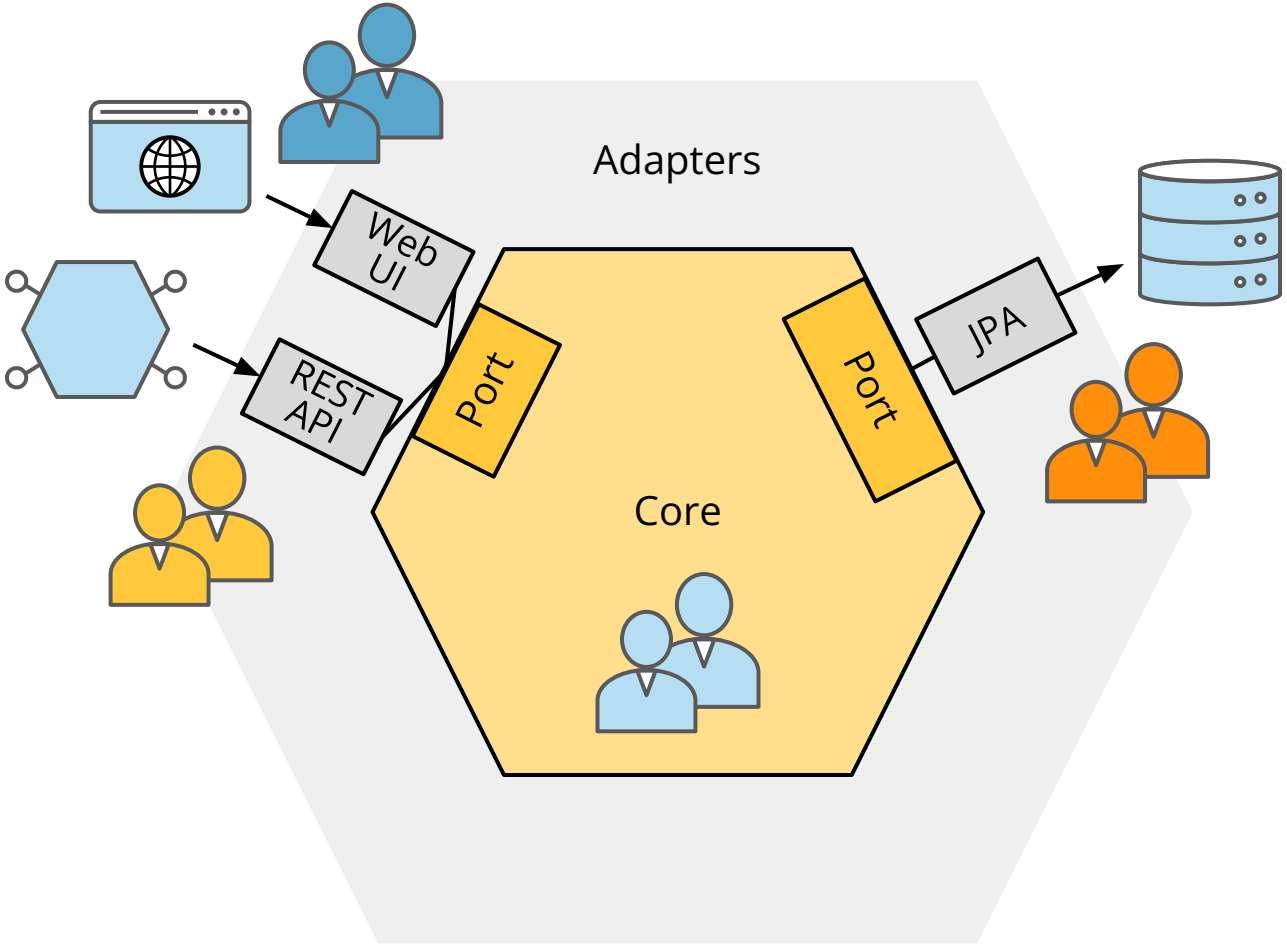




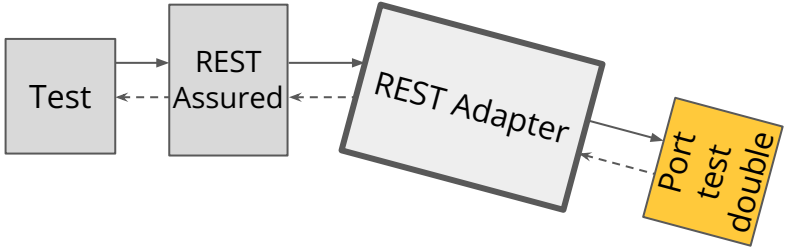


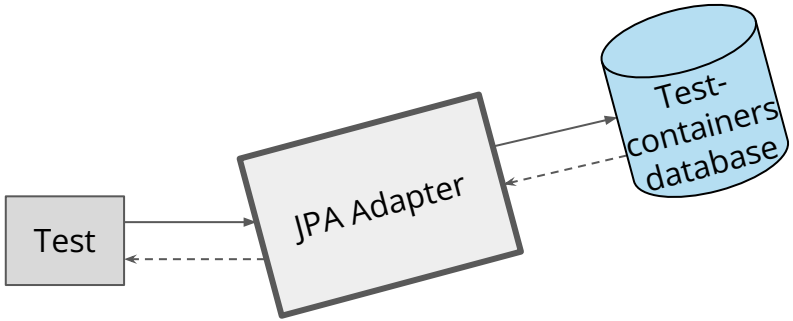


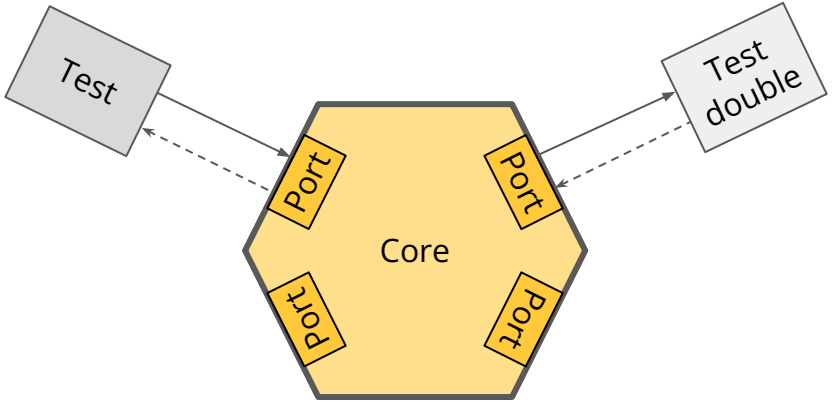
Goal #3: Independent Development



Goal #4: Independent Testability

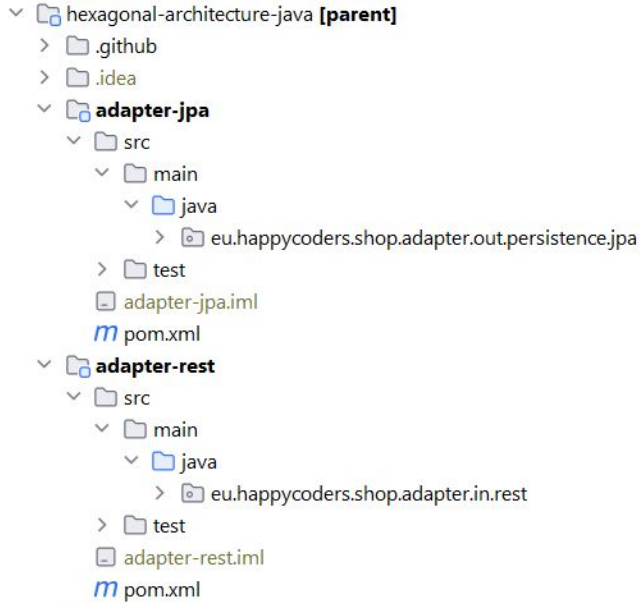


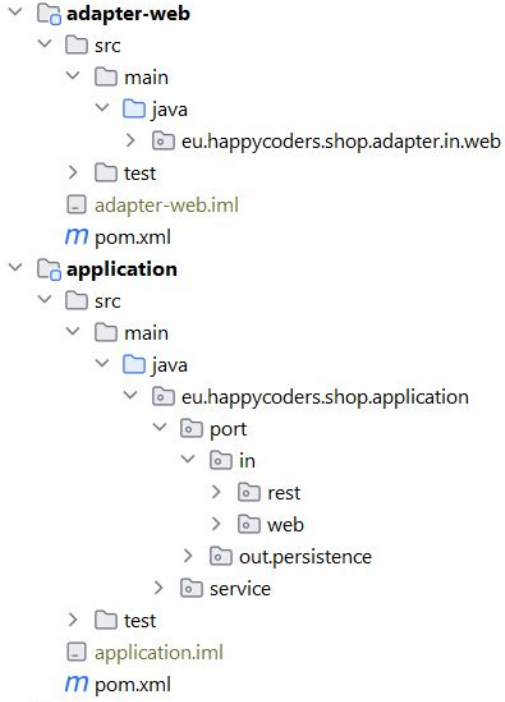


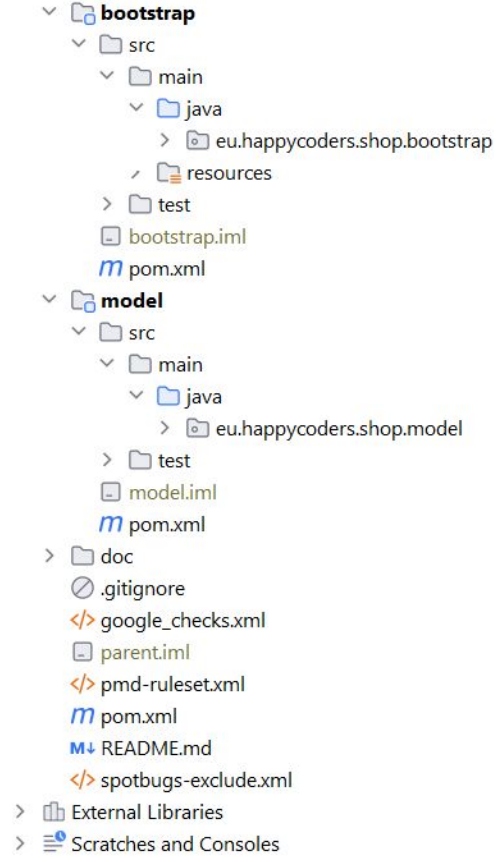


Hexagonal Architecture

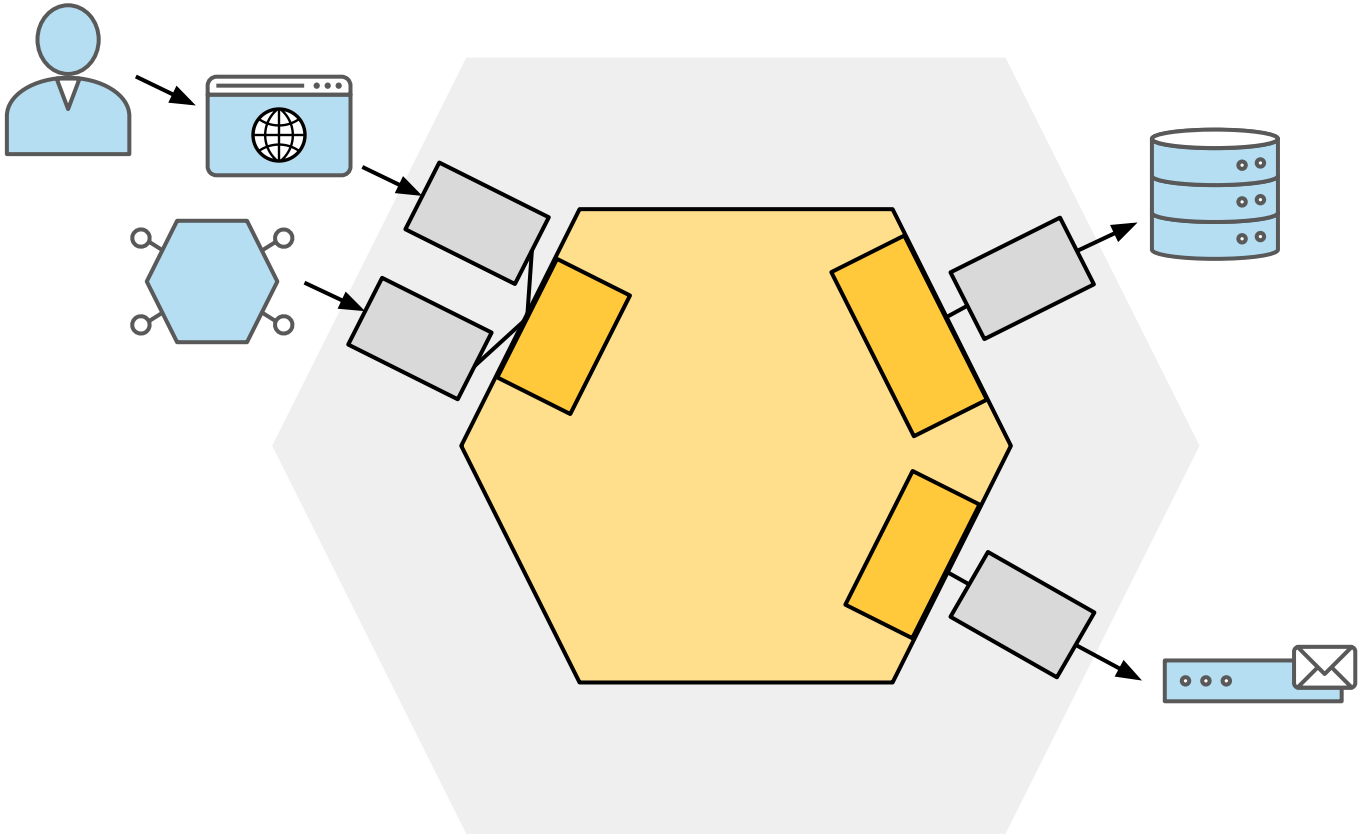
- ✔ Changeability
- ✔ Loose Coupling
- ✔ Independent Development
- ✔ Independent Testability







End



Links, Slides

sven@happycoders.eu

https://www.happycoders.eu/

https://linkedin.com/in/sven-woltmann/

https://youtube.com/c/happycoders

LinkedIn