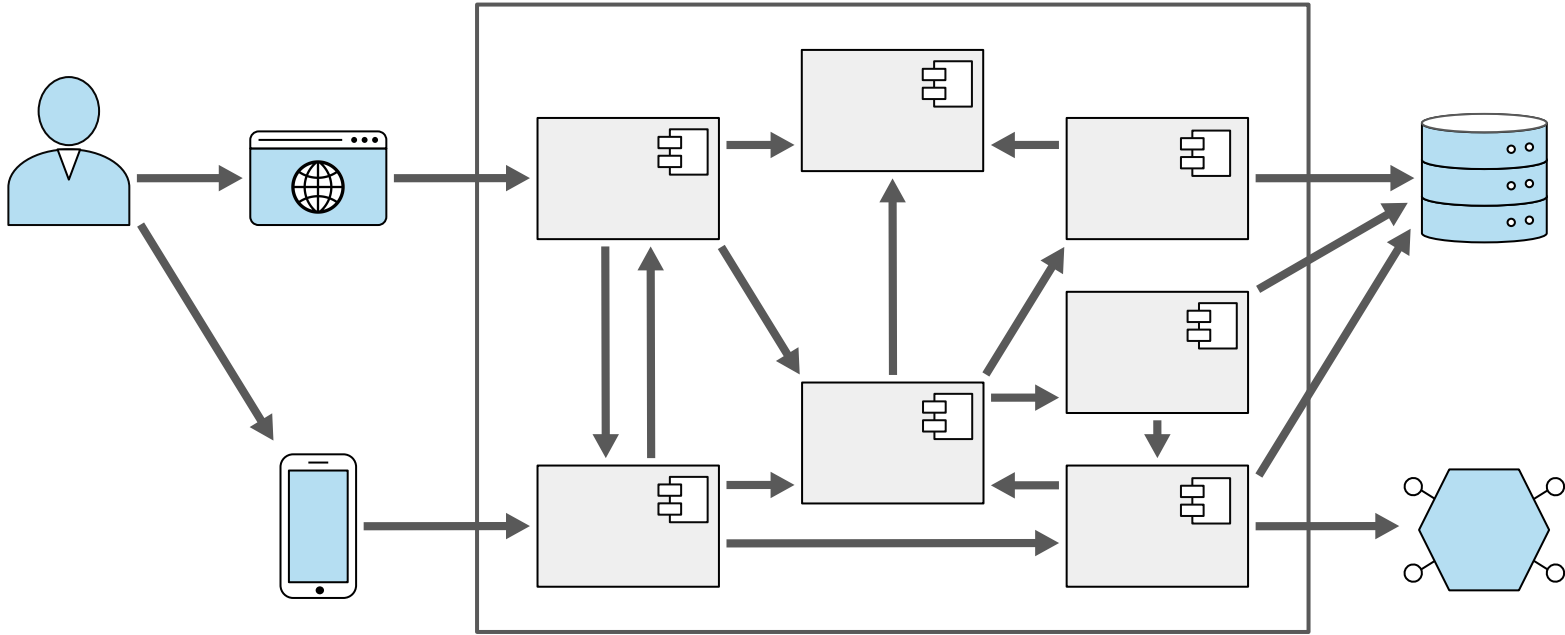


Hexagonale Architektur: Robuste Software dank Schnittstellen statt Schichten

Sven Woltmann

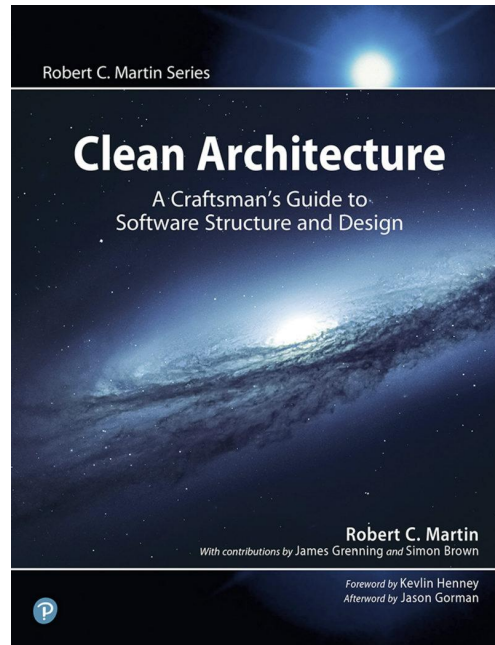
Was ist Softwarearchitektur?



Was ist das Ziel von Softwarearchitektur?

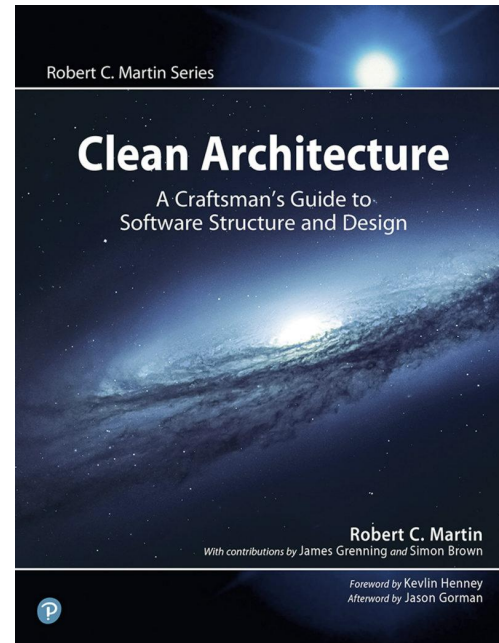
“The goal of a software architecture is to minimize the human resources required to build and maintain the required system.”

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



Was ist das Ziel von Softwarearchitektur?

„Das Ziel einer Software-Architektur ist es, den Personalaufwand für die Erstellung und Wartung des gewünschten Systems zu minimieren.“



Was ist das Ziel von Softwarearchitektur?

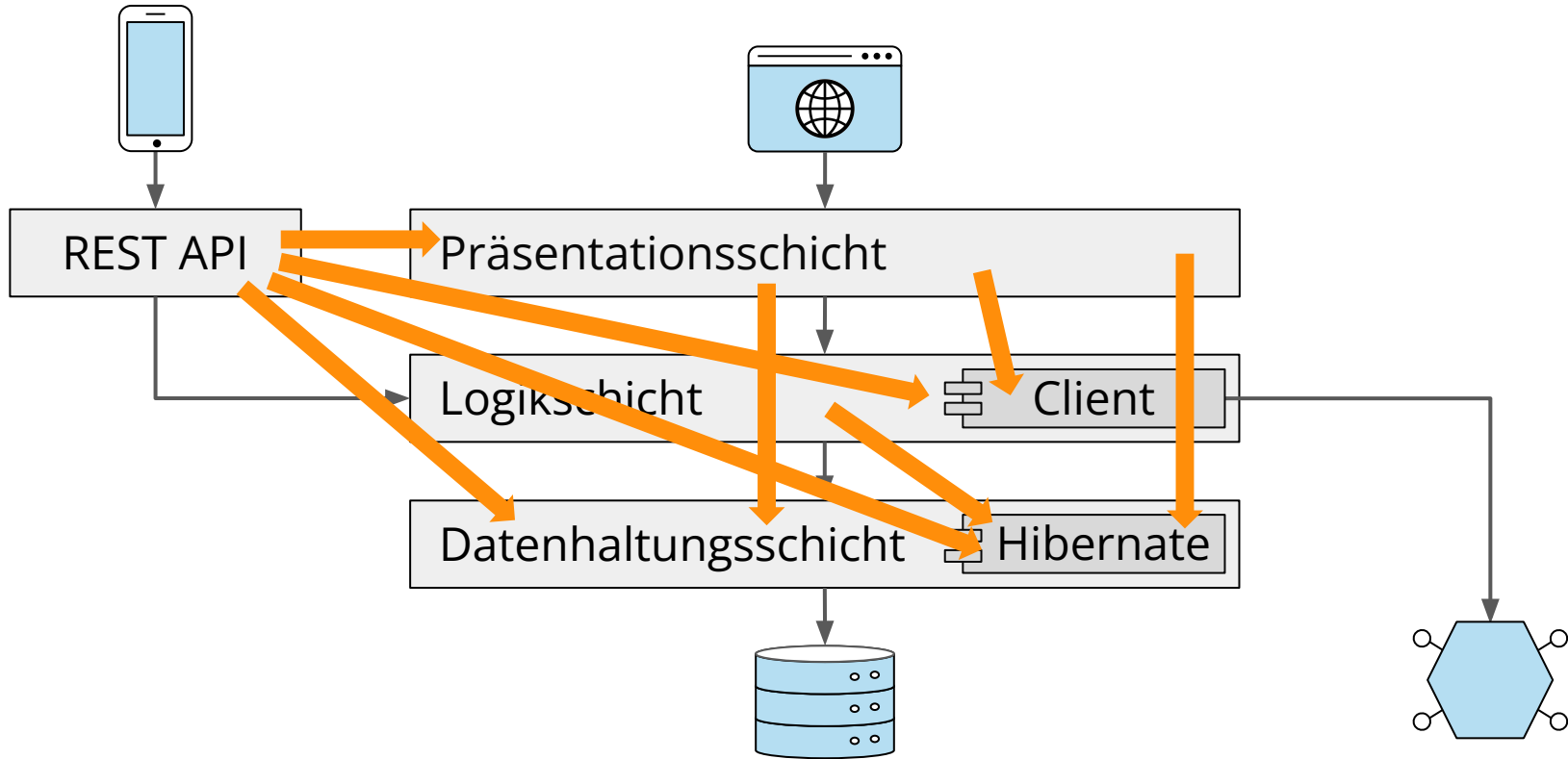
Eine gute Architektur erlaubt es, Software während ihrer Lebensdauer mit möglichst geringem, gleichbleibendem Aufwand zu ändern.

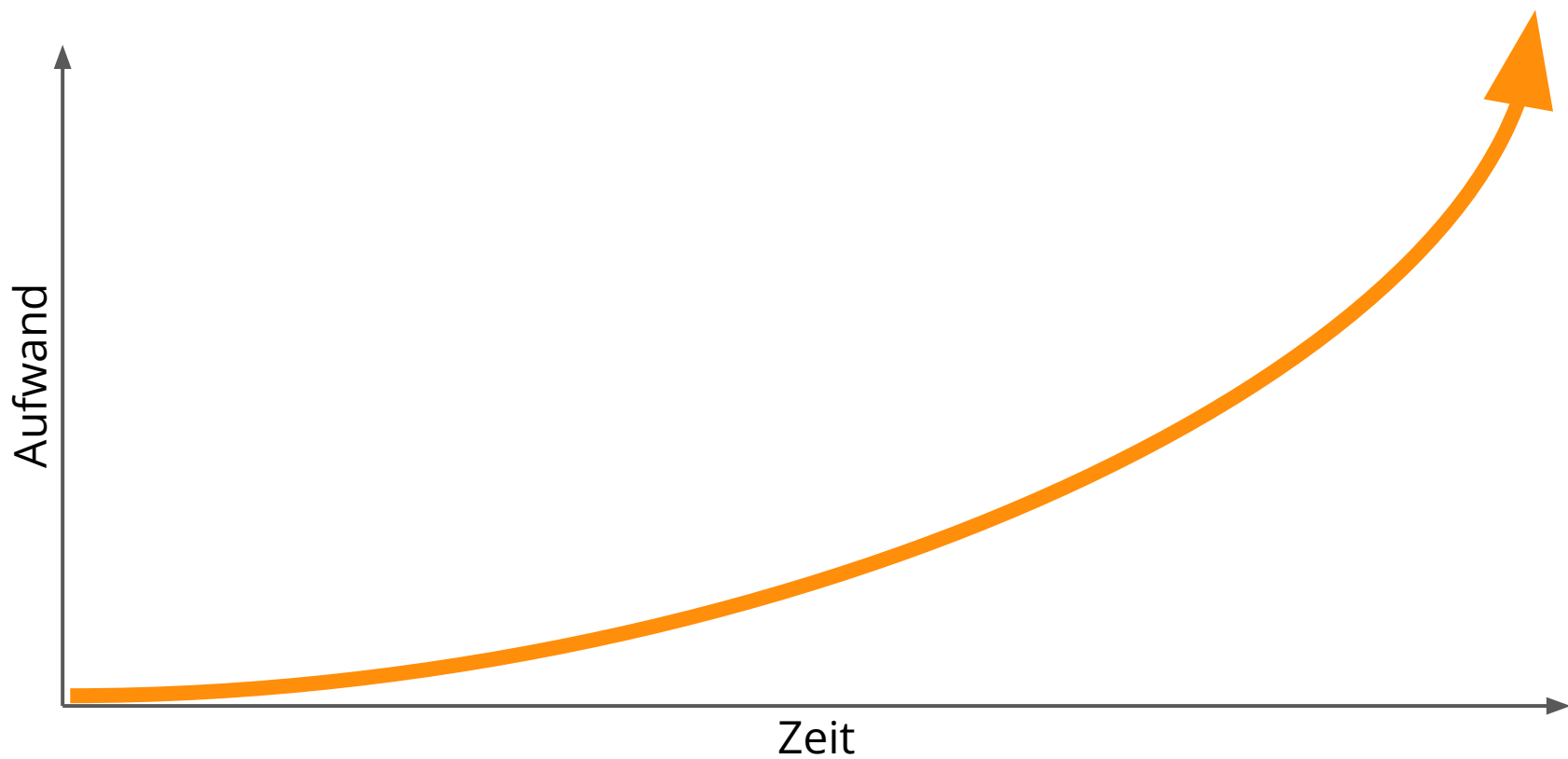
“To keep software soft.”

— Robert C. Martin



Schichtenarchitektur





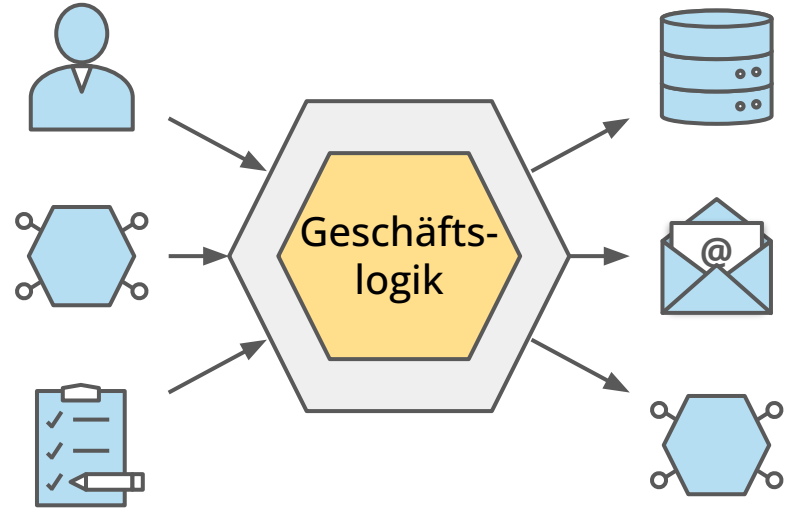
Hexagonale Architektur

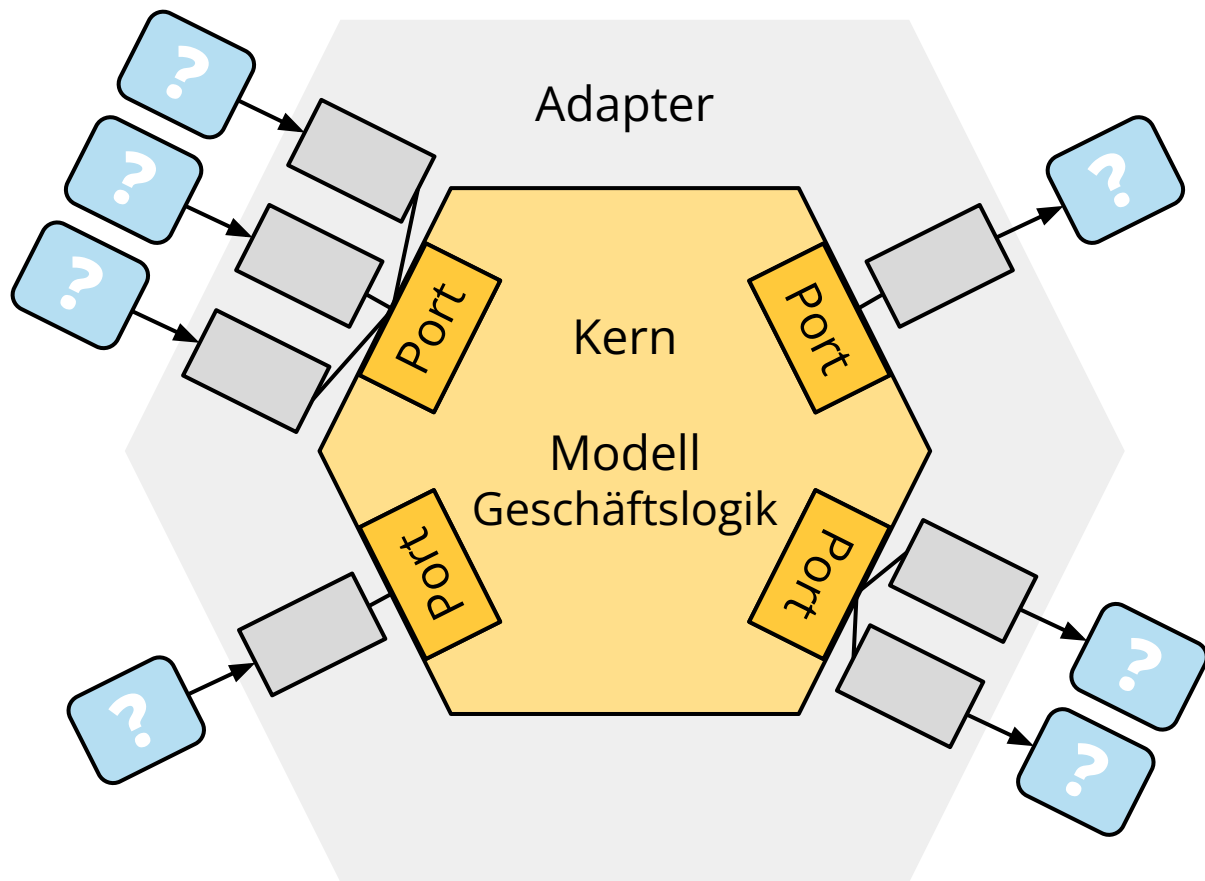
<https://alistair.cockburn.us/hexagonal-architecture/>

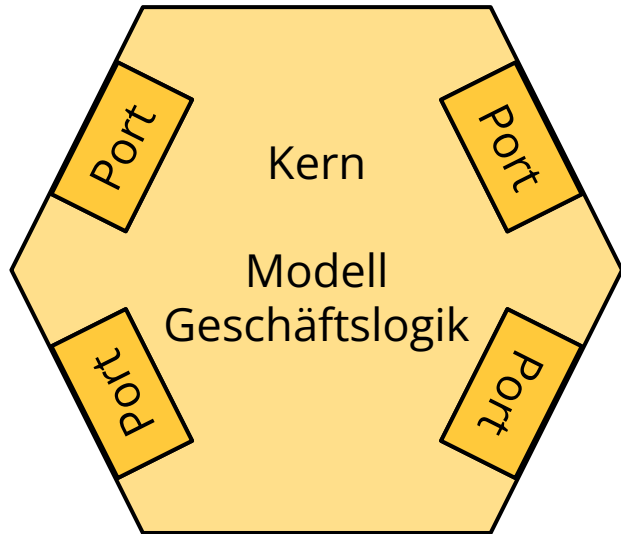


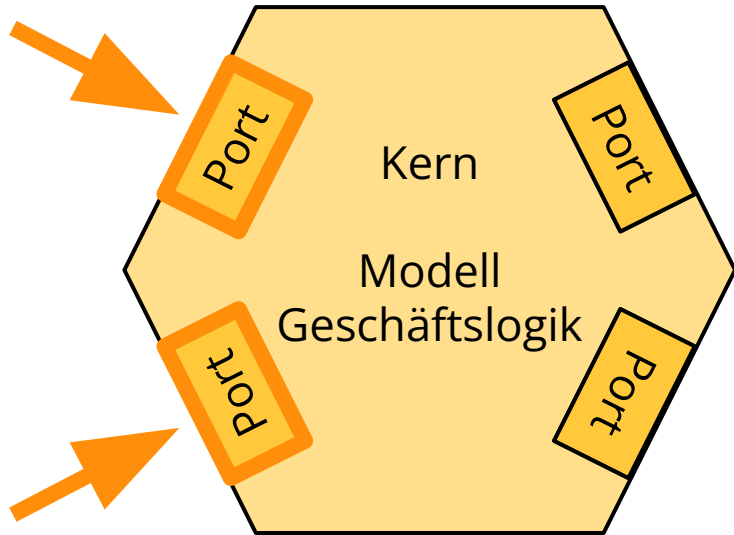
Ziele

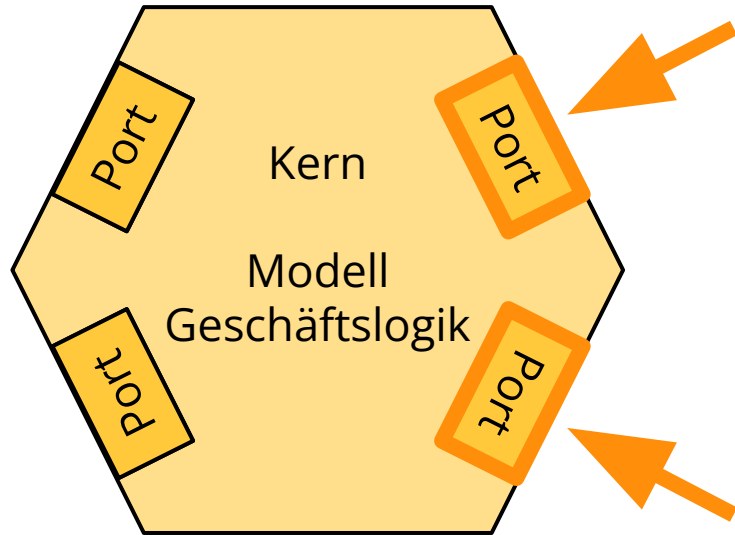
1. Isolierung von *steuernder* Infrastruktur
2. Isolierung von *gesteuerter* Infrastruktur
3. Modernisierung von Infrastruktur ohne Anpassung der Geschäftslogik

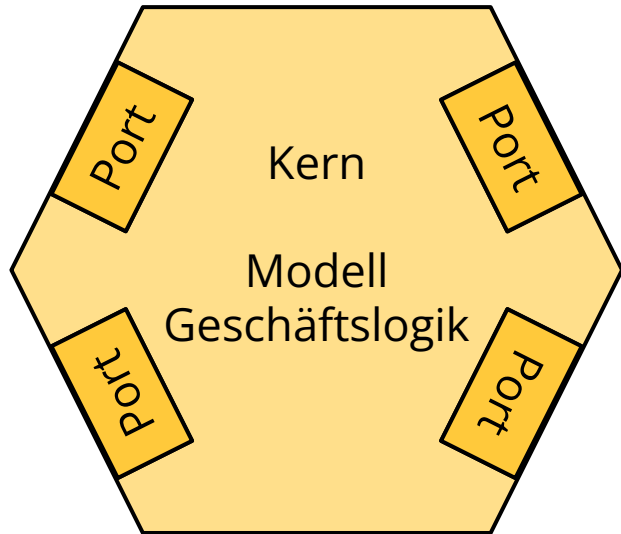


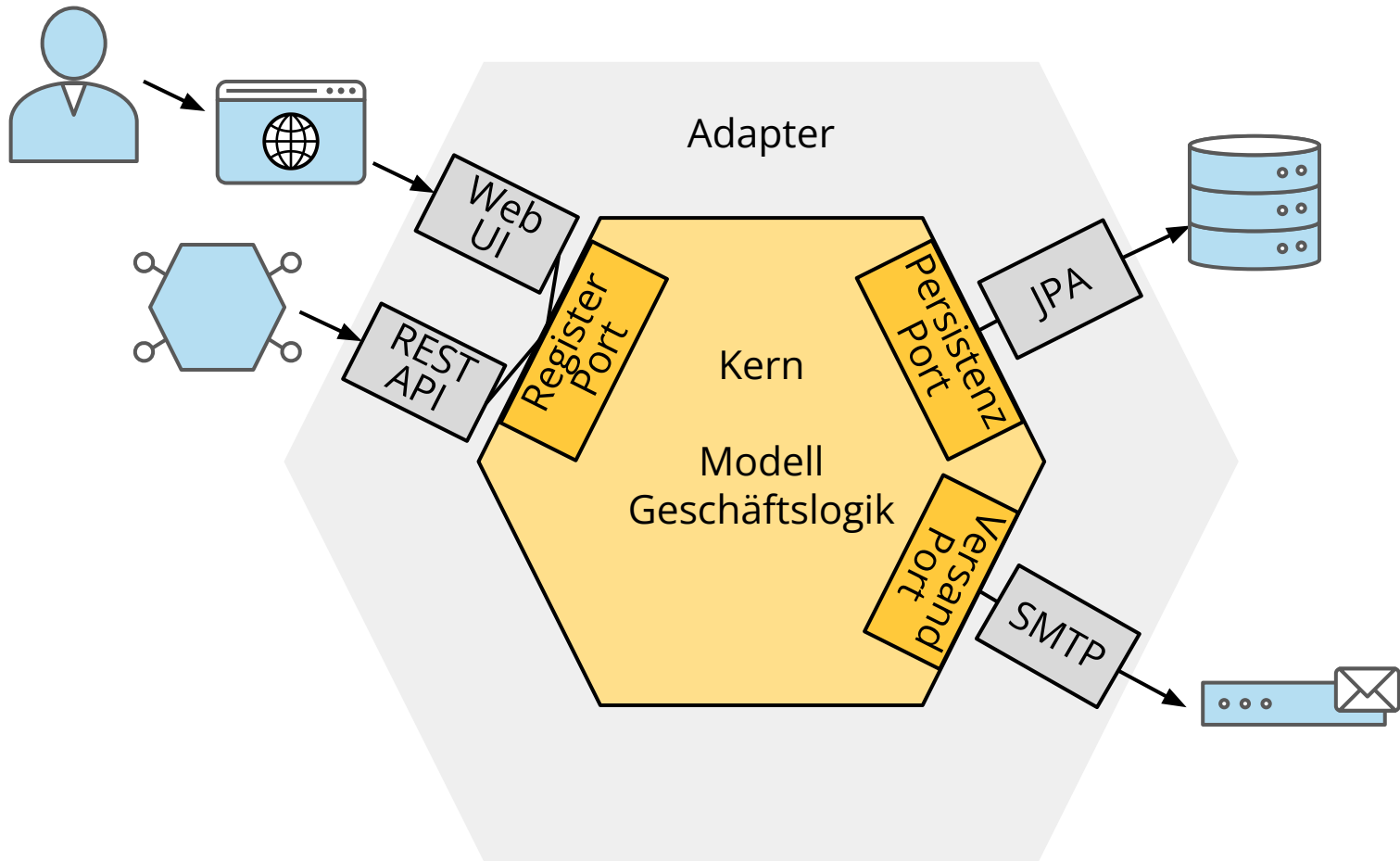


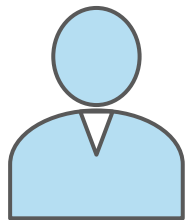




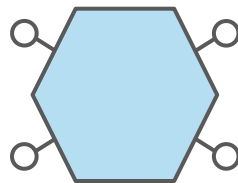








Ausfüllen
Absenden



```
{  "name": "Sven",  "email": "...",}
```

name=Sven&email=sven@happycoders.eu

Web UI
Adapter

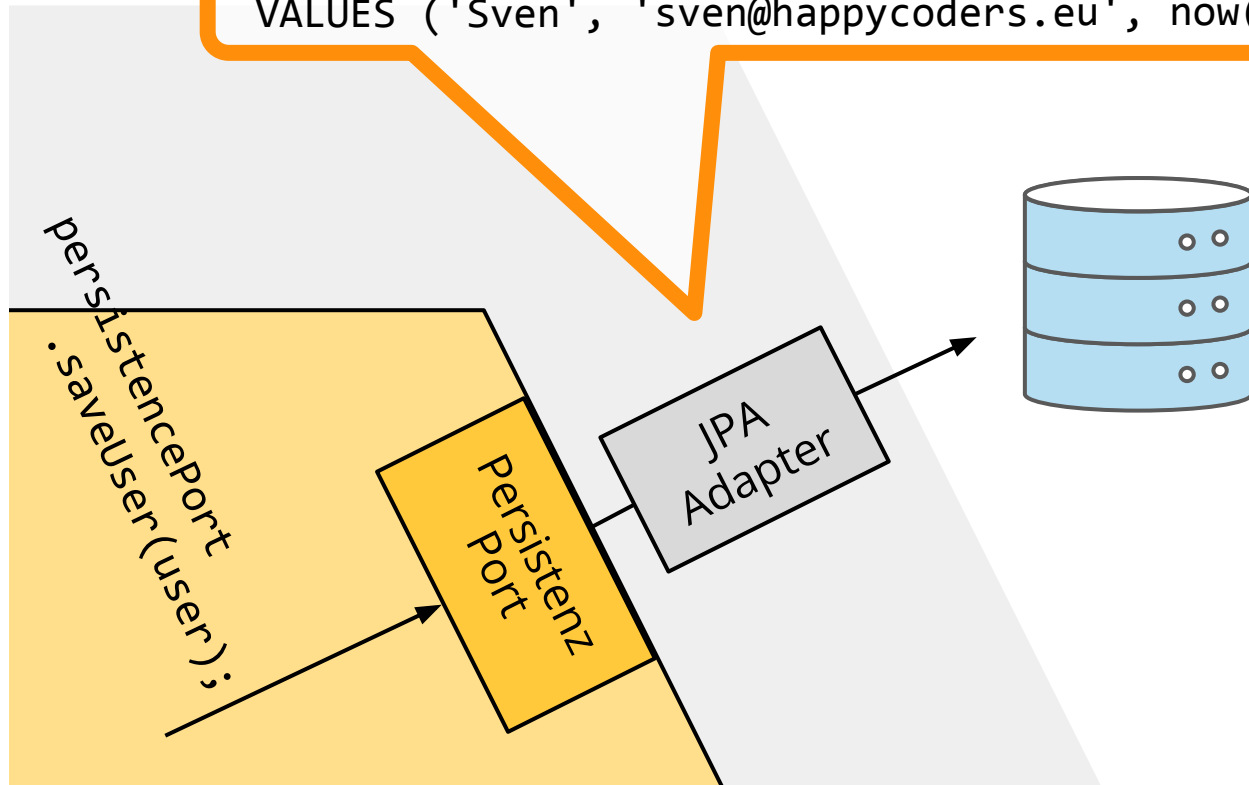
REST API
Adapter

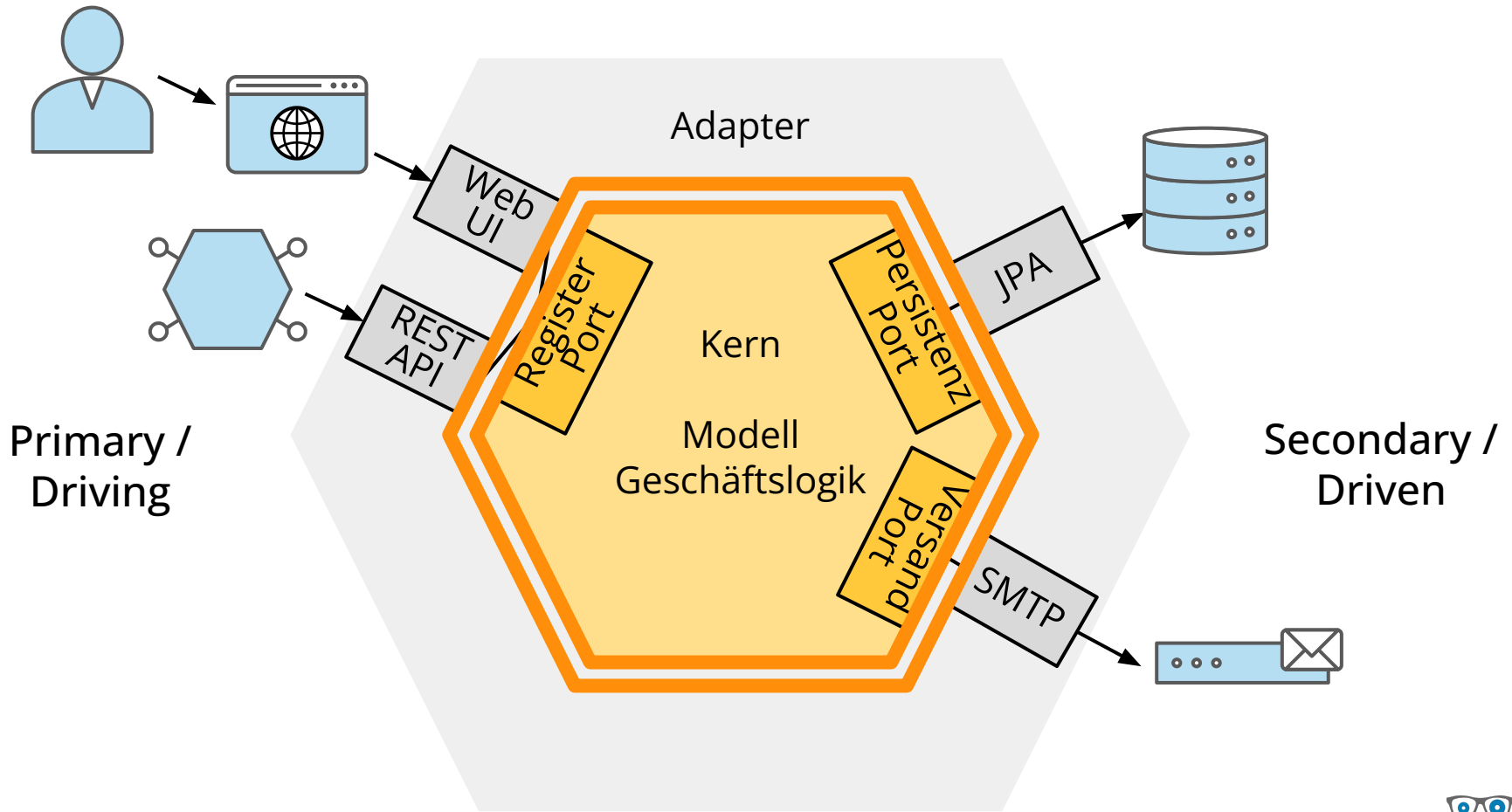
Register
port

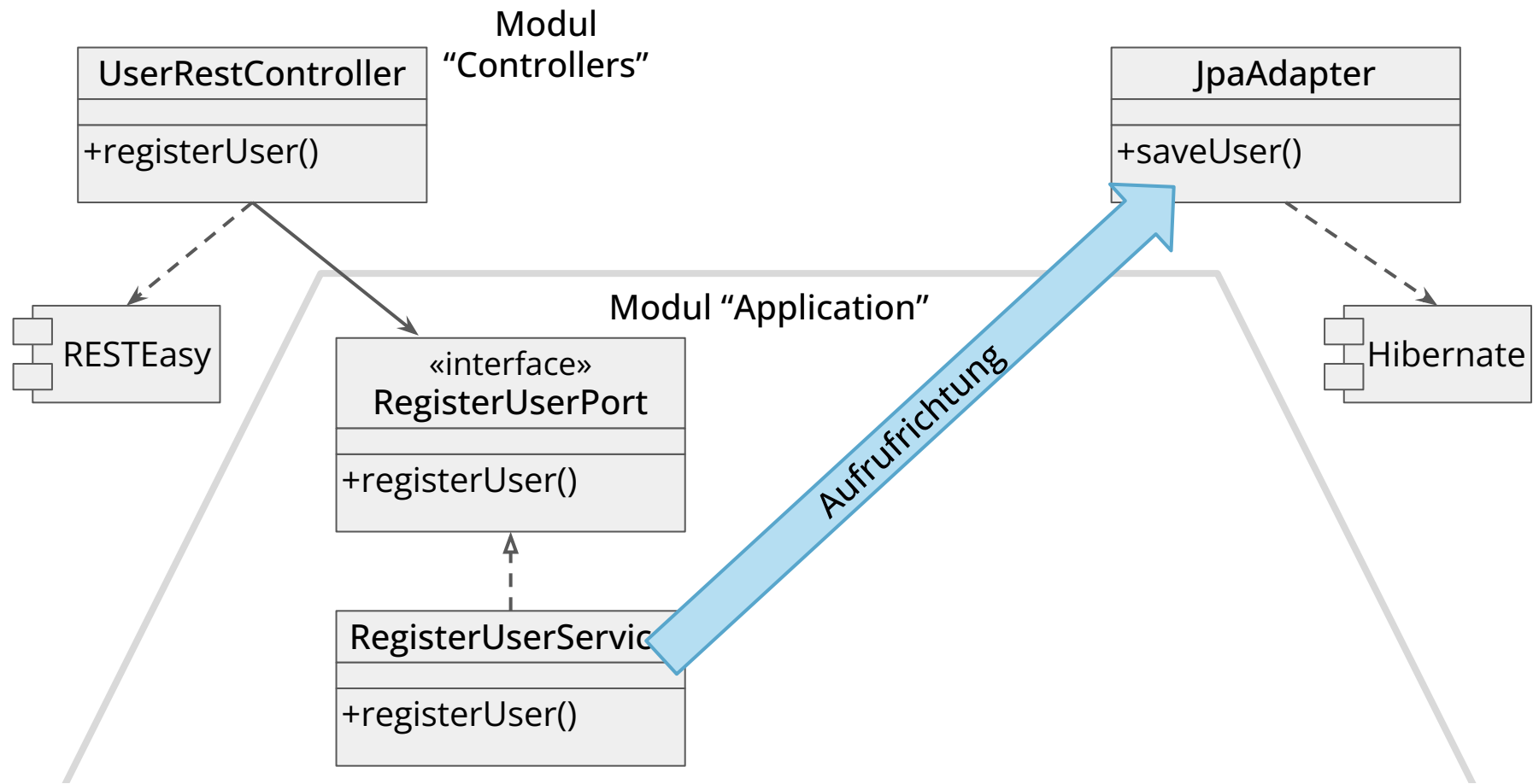
registerPort.registerUser(
 "Sven",
 "sven@happycoders.eu");



```
INSERT INTO User (name, email, registrationDate)  
VALUES ('Sven', 'sven@happycoders.eu', now());
```





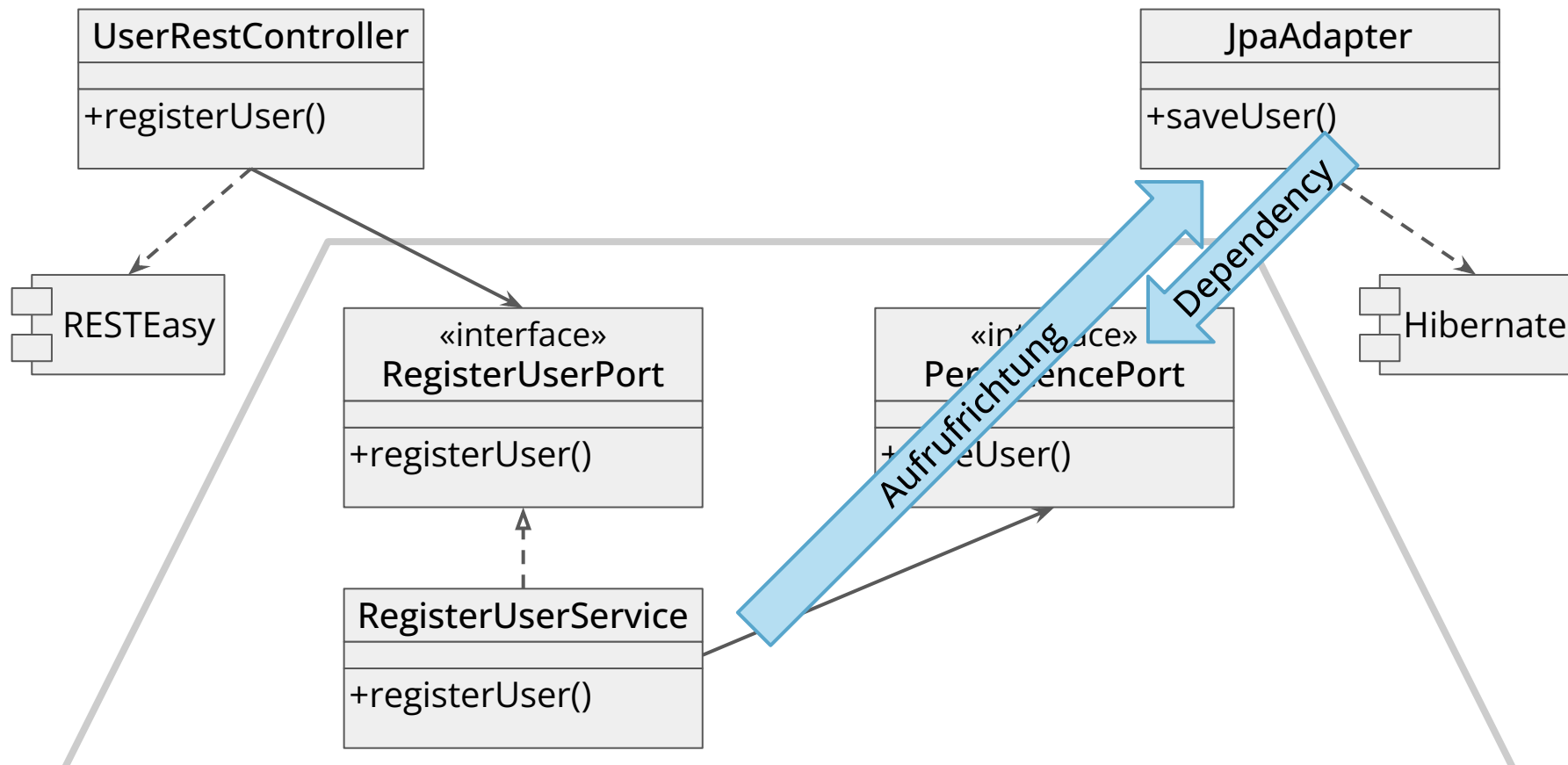


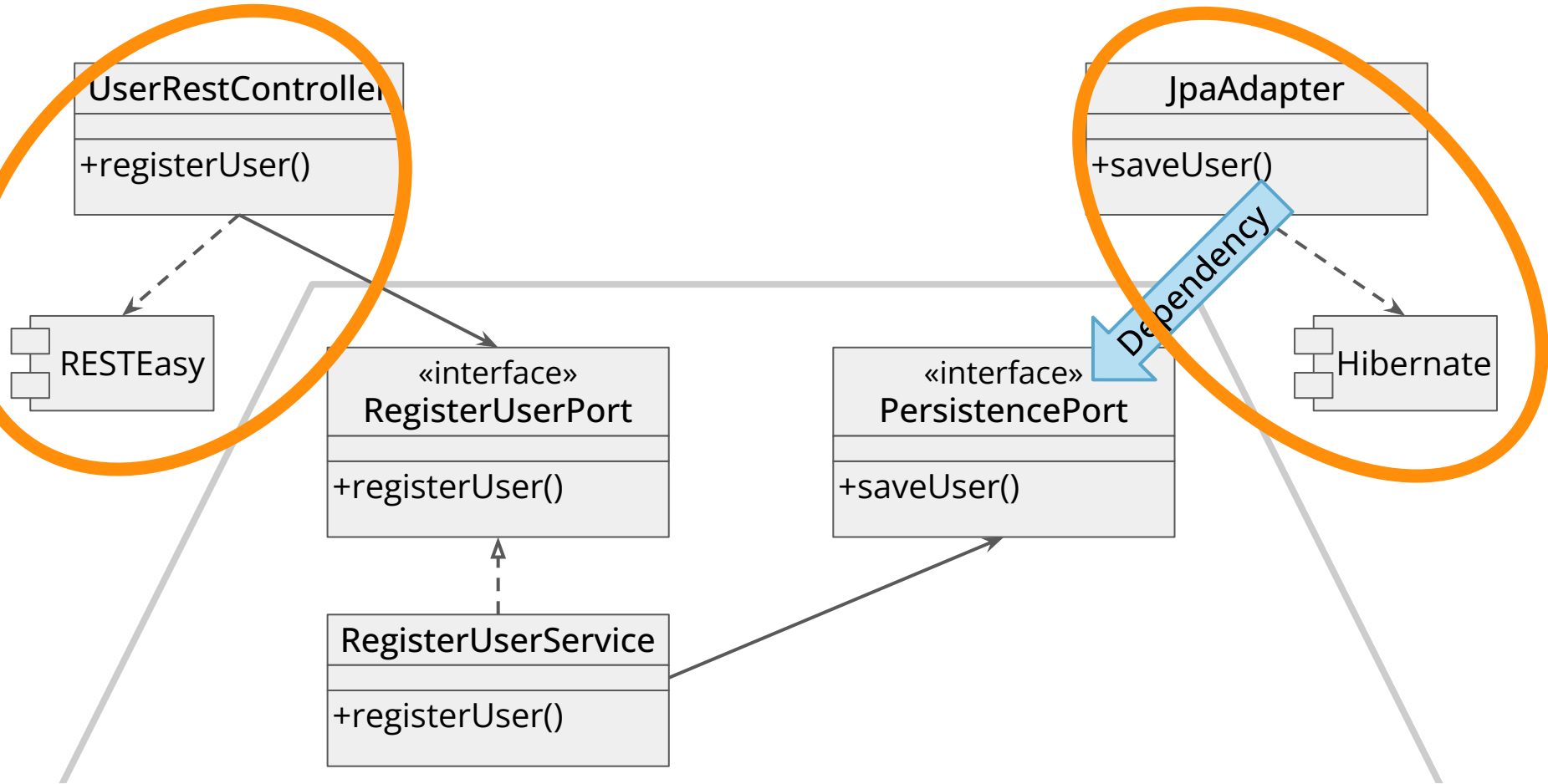
Dependency Inversion Principle



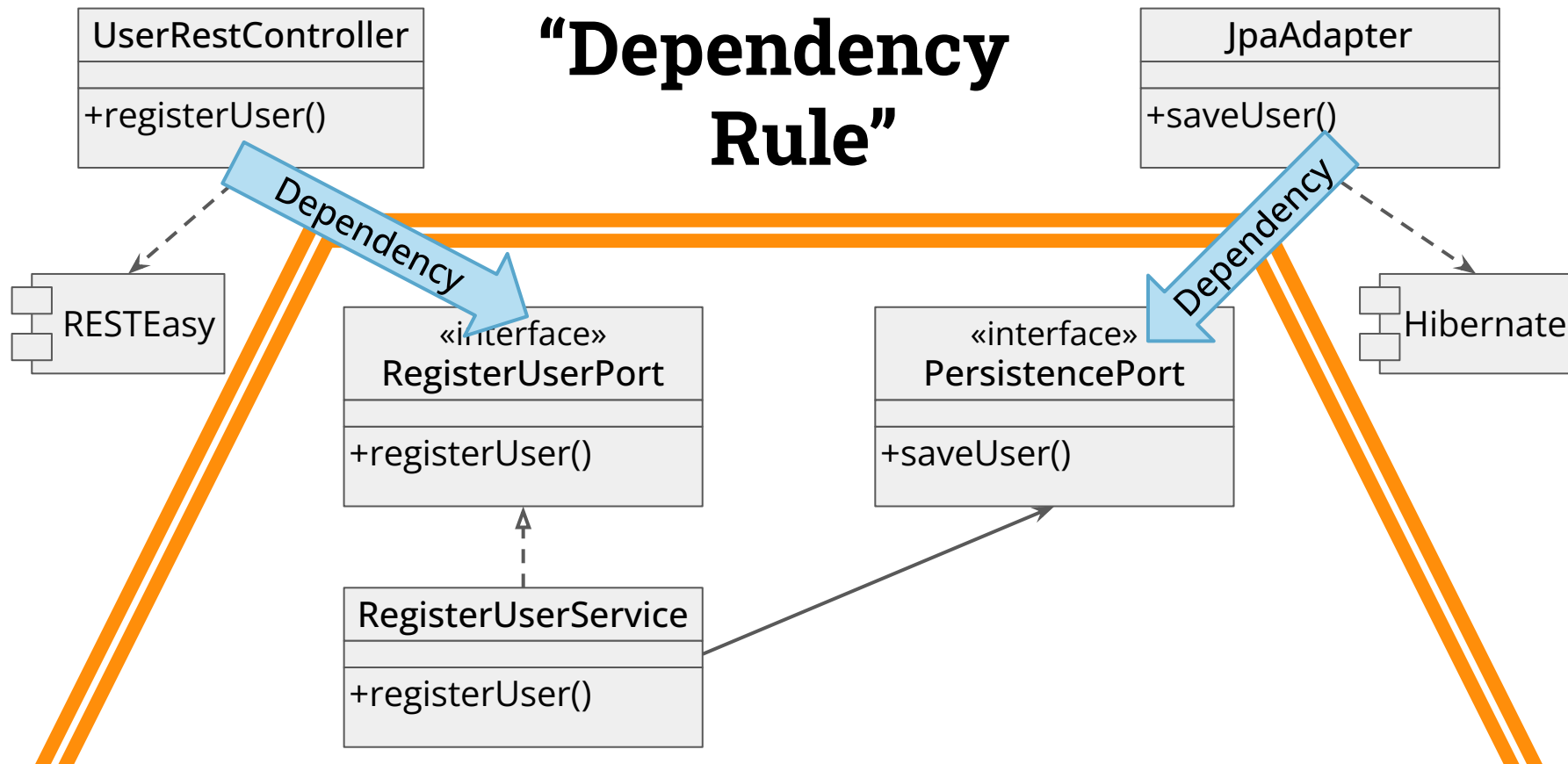
SOLID

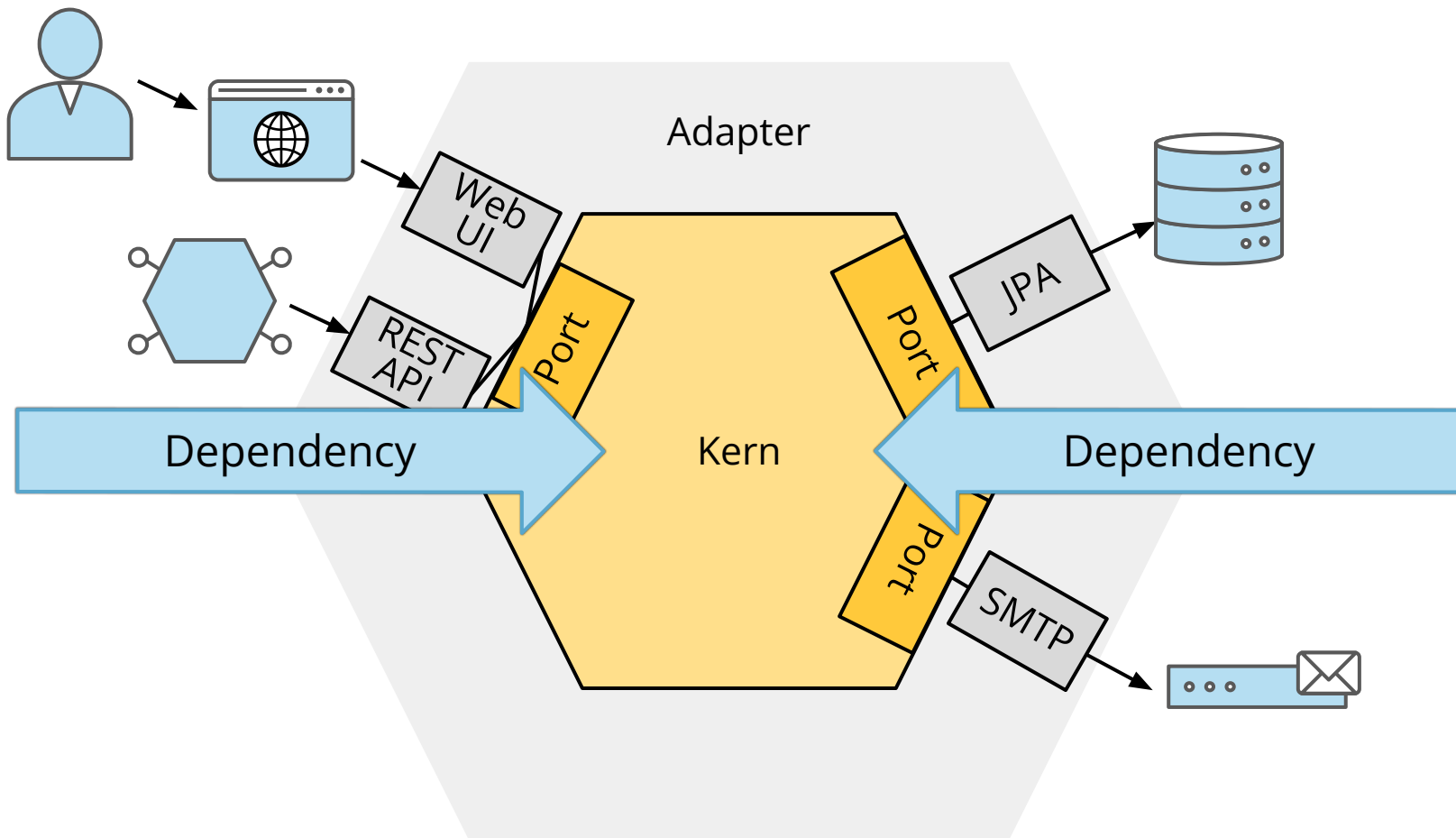






“Dependency Rule”





Mapping



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

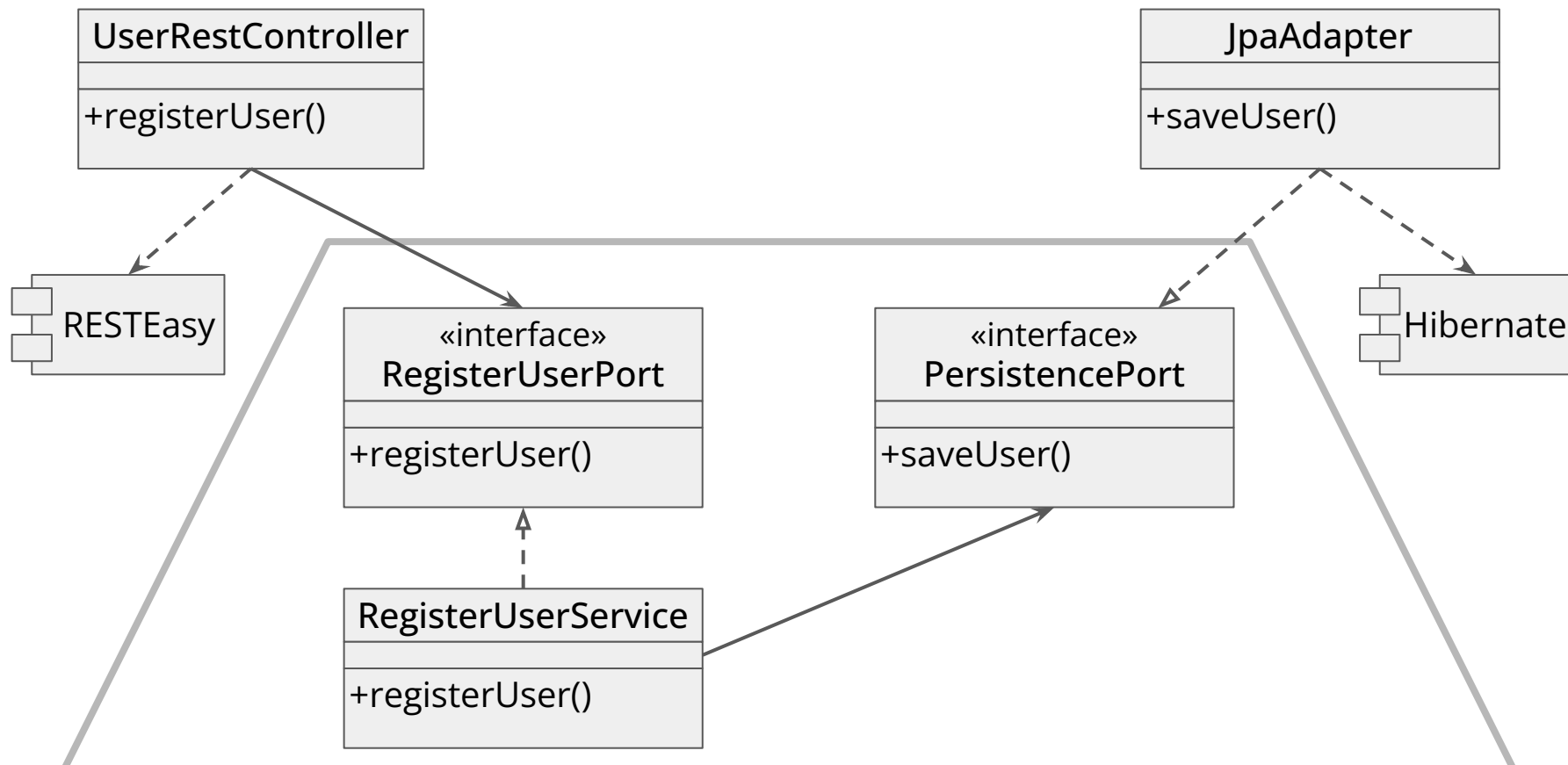
@Entity
public class User {

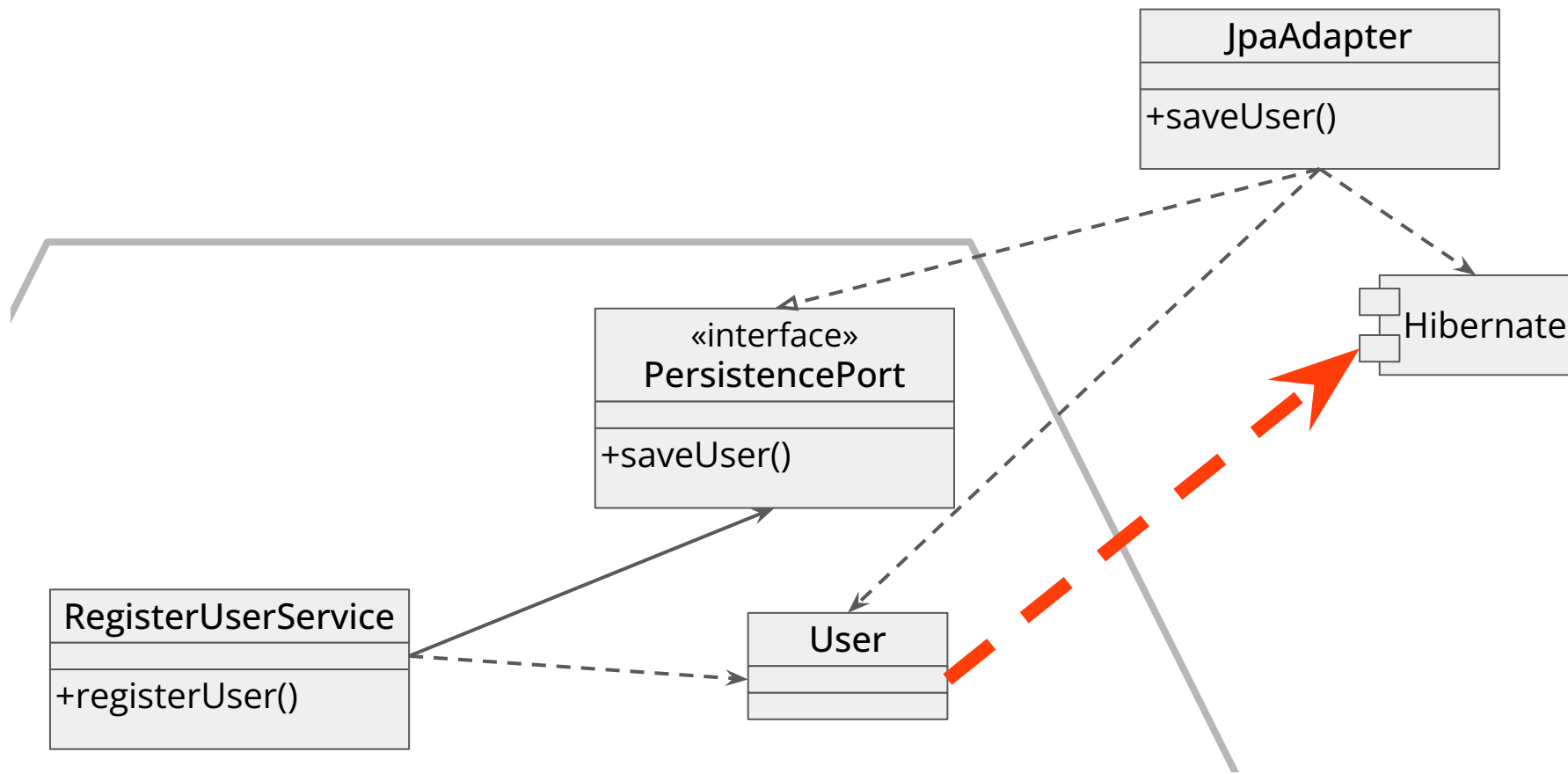
    @Id private Long id;

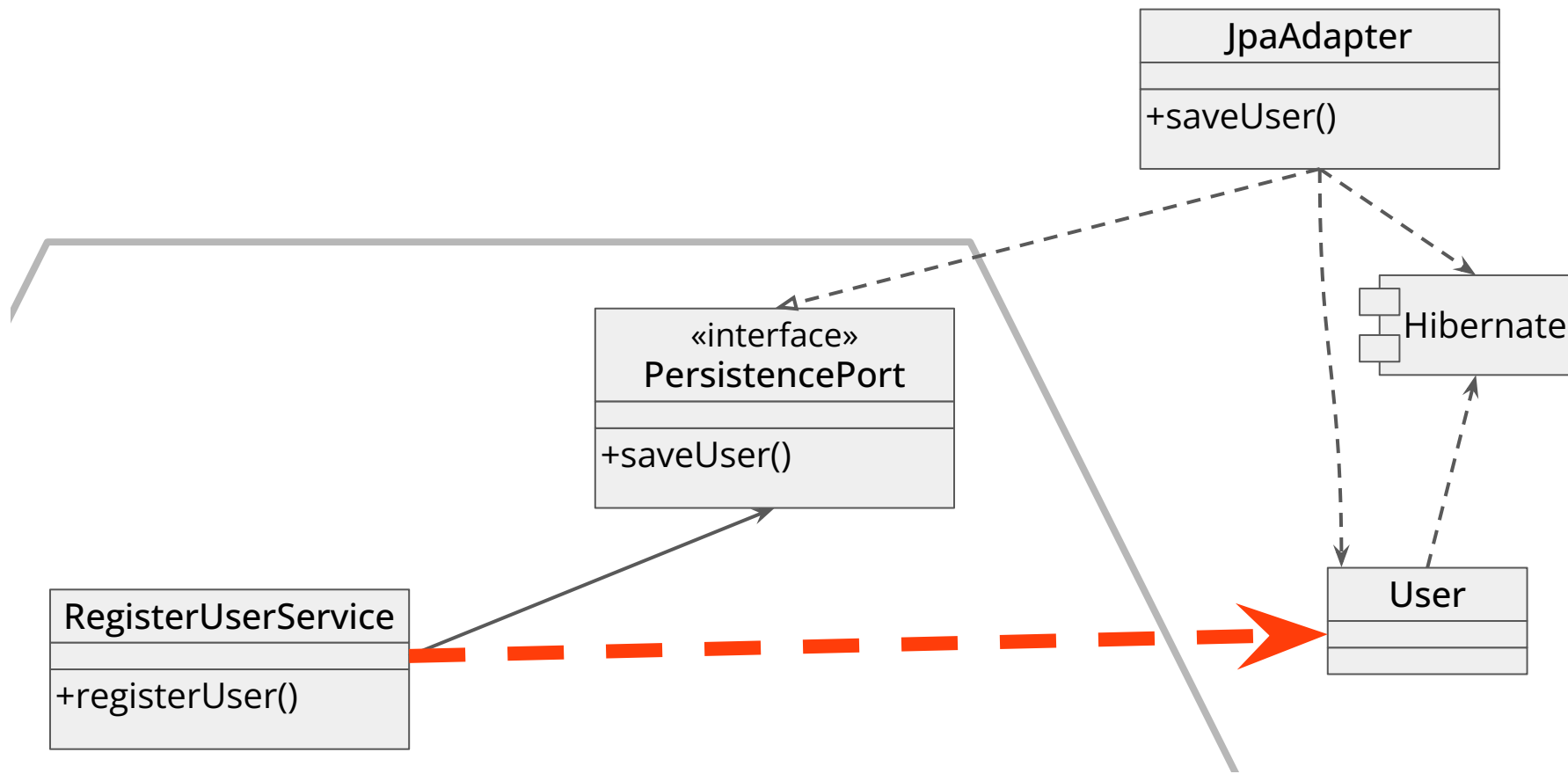
    @Column(nullable = false) private String name;
    @Column(nullable = false) private String email;
    @Column(nullable = false) private Instant registeredAt;

    // ...
}
```







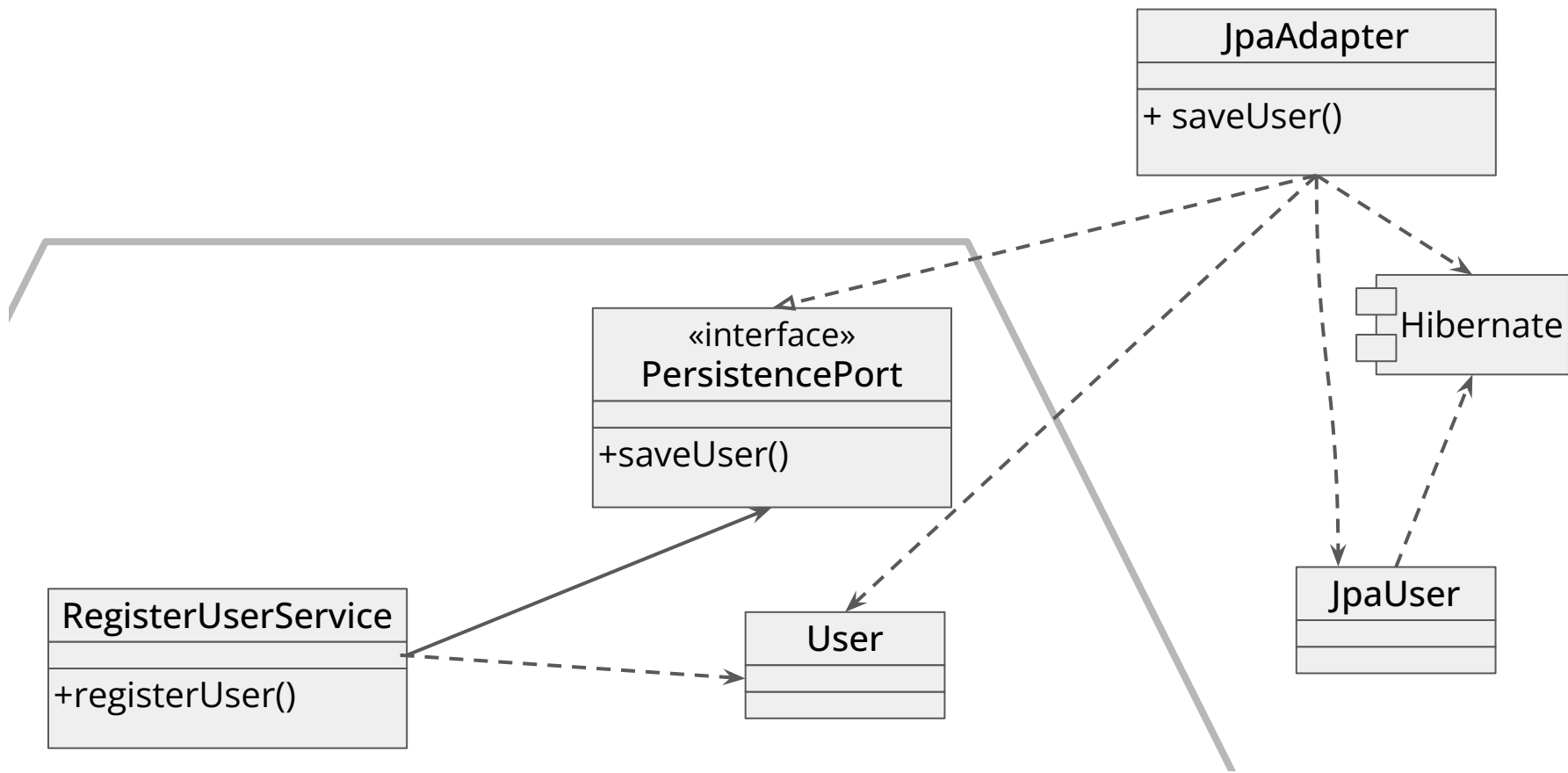


Zwei-Wege-Mapping



Two-Way Mapping





```
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;  
  
    // ...  
}
```



```
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;
```

```
    // ...
```

```
}
```

```
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;
```

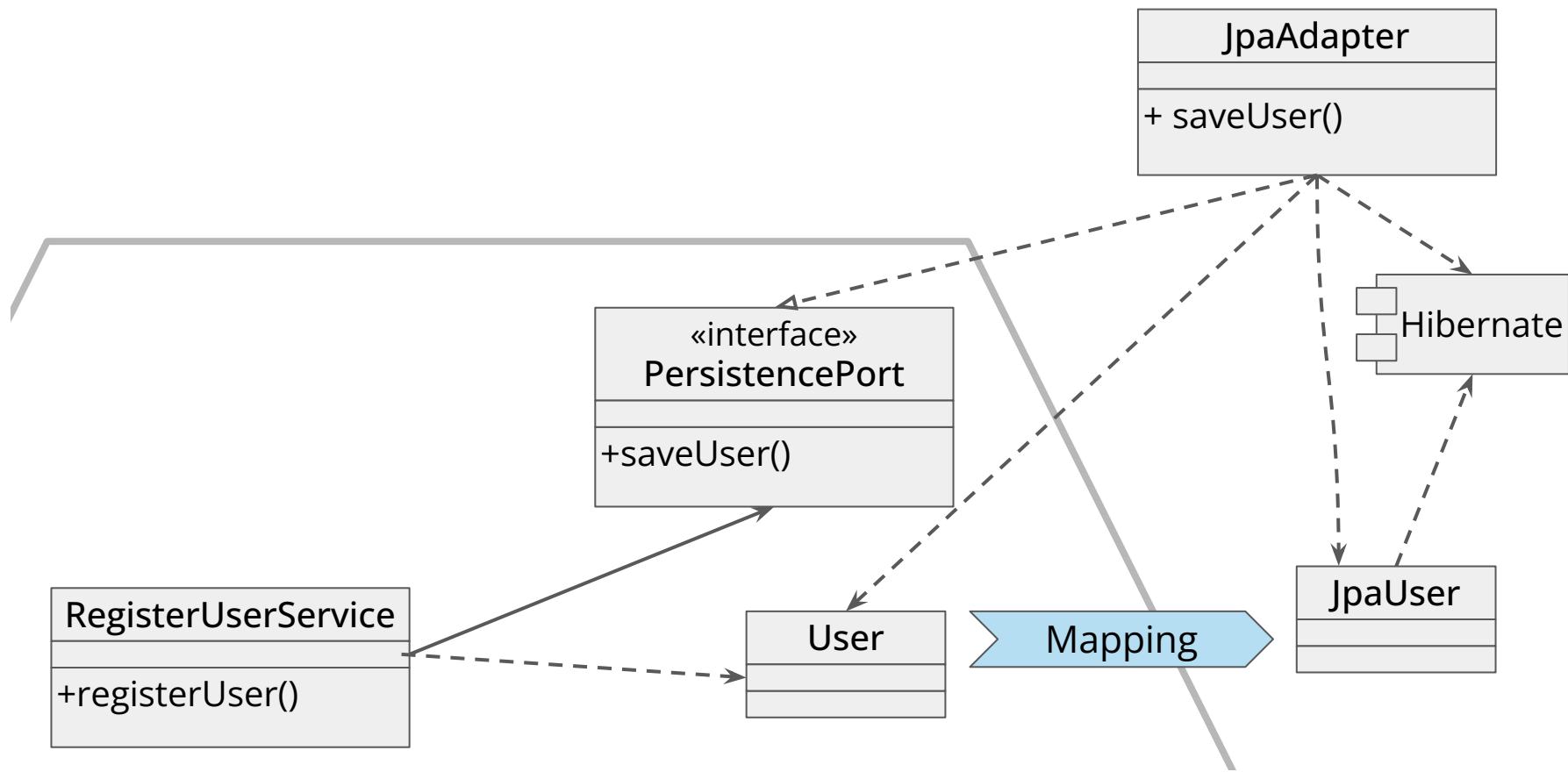
```
        this.registeredAt = Instant.now();
```

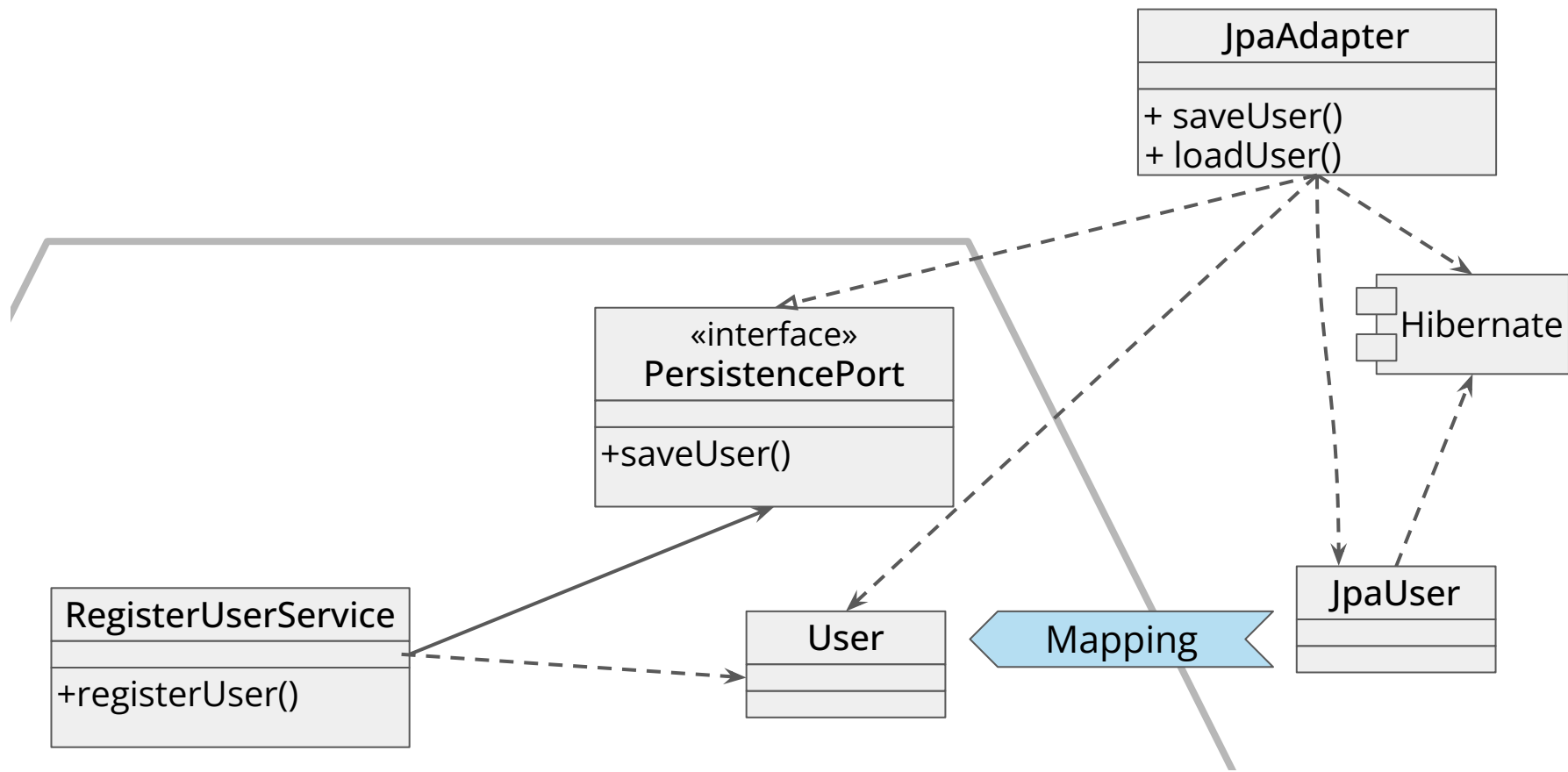
```
    }
```

```
    // ...
```

```
}
```







```
public class JpaAdapter implements PersistencePort {

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

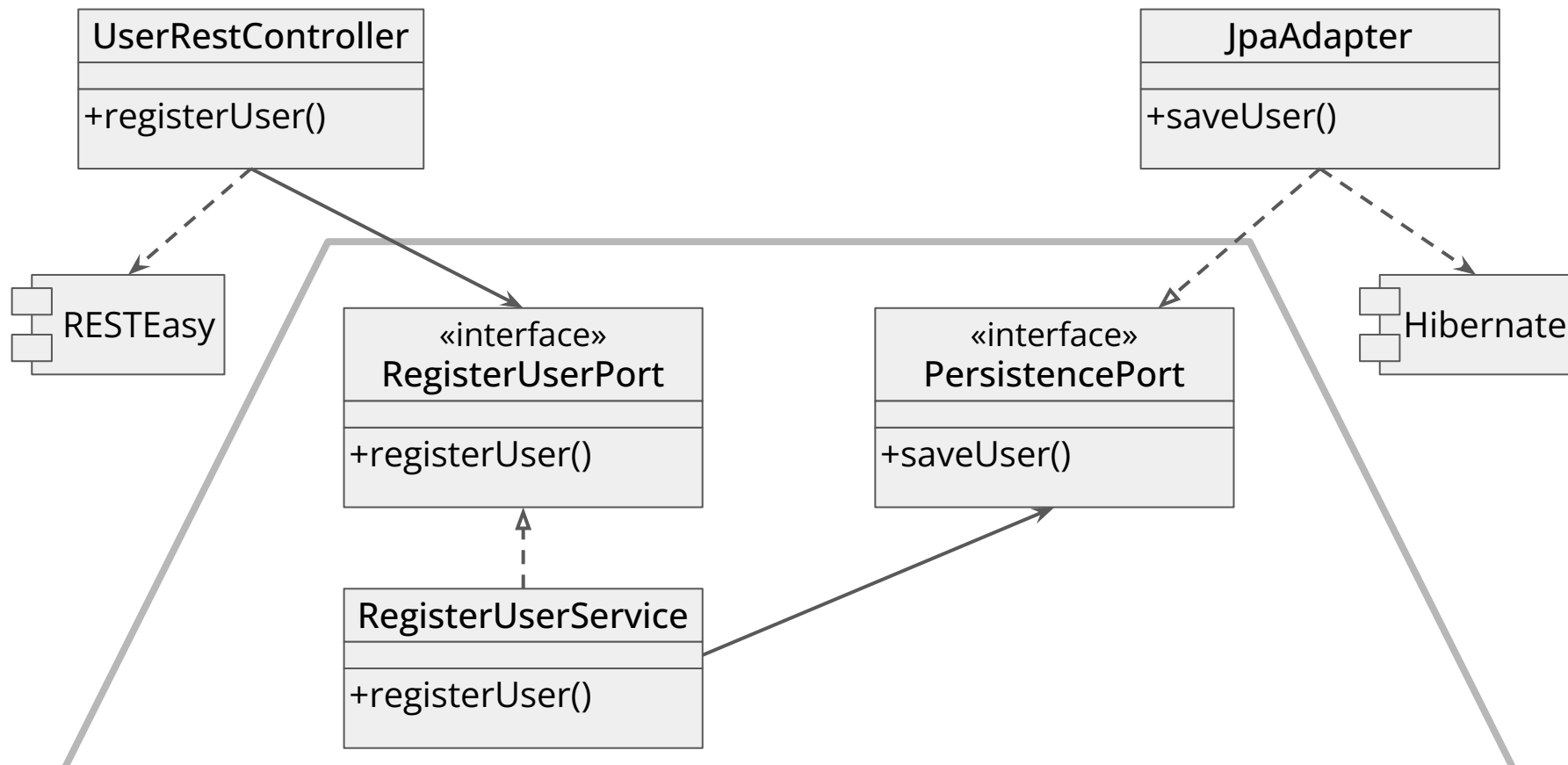
    @Override
    @Transactional
    public Optional<User> loadUser(long userId) {
        Optional<JpaUser> jpaUser = repository.findById(userId);
        return jpaUser.map(mapper::toUser);
    }

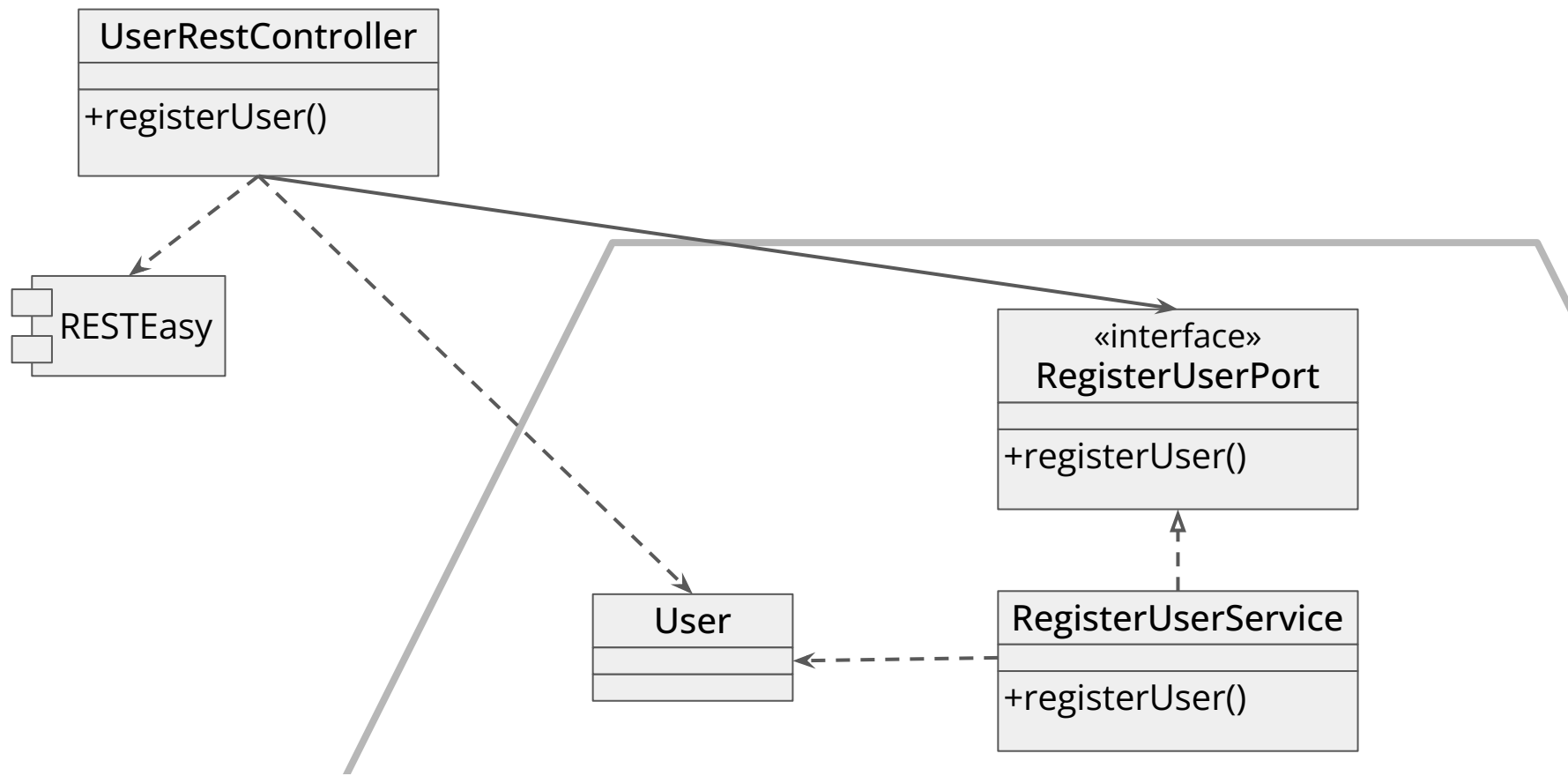
}
```



```
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);  
    }  
  
}
```







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

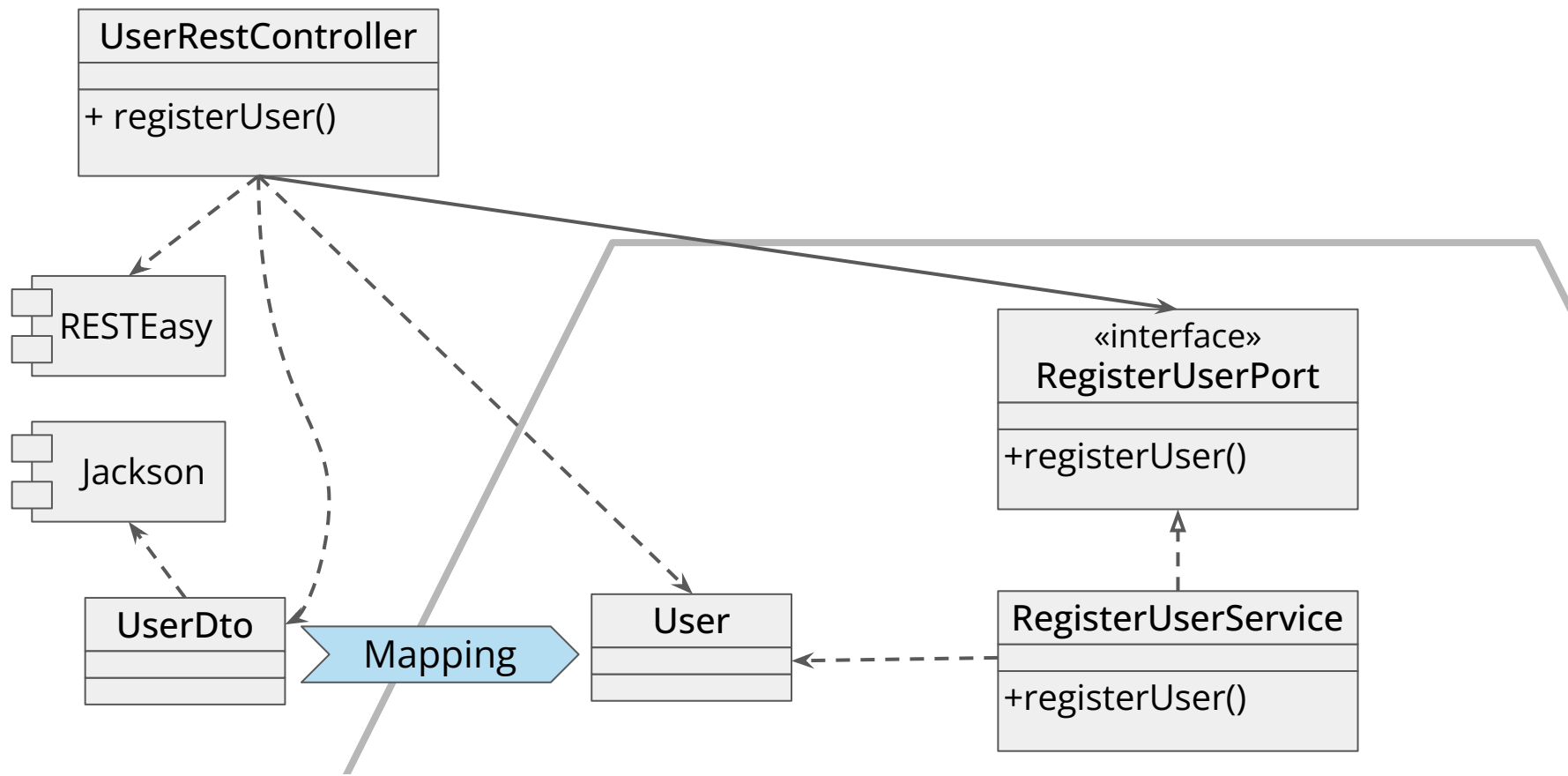
public class UserDto {

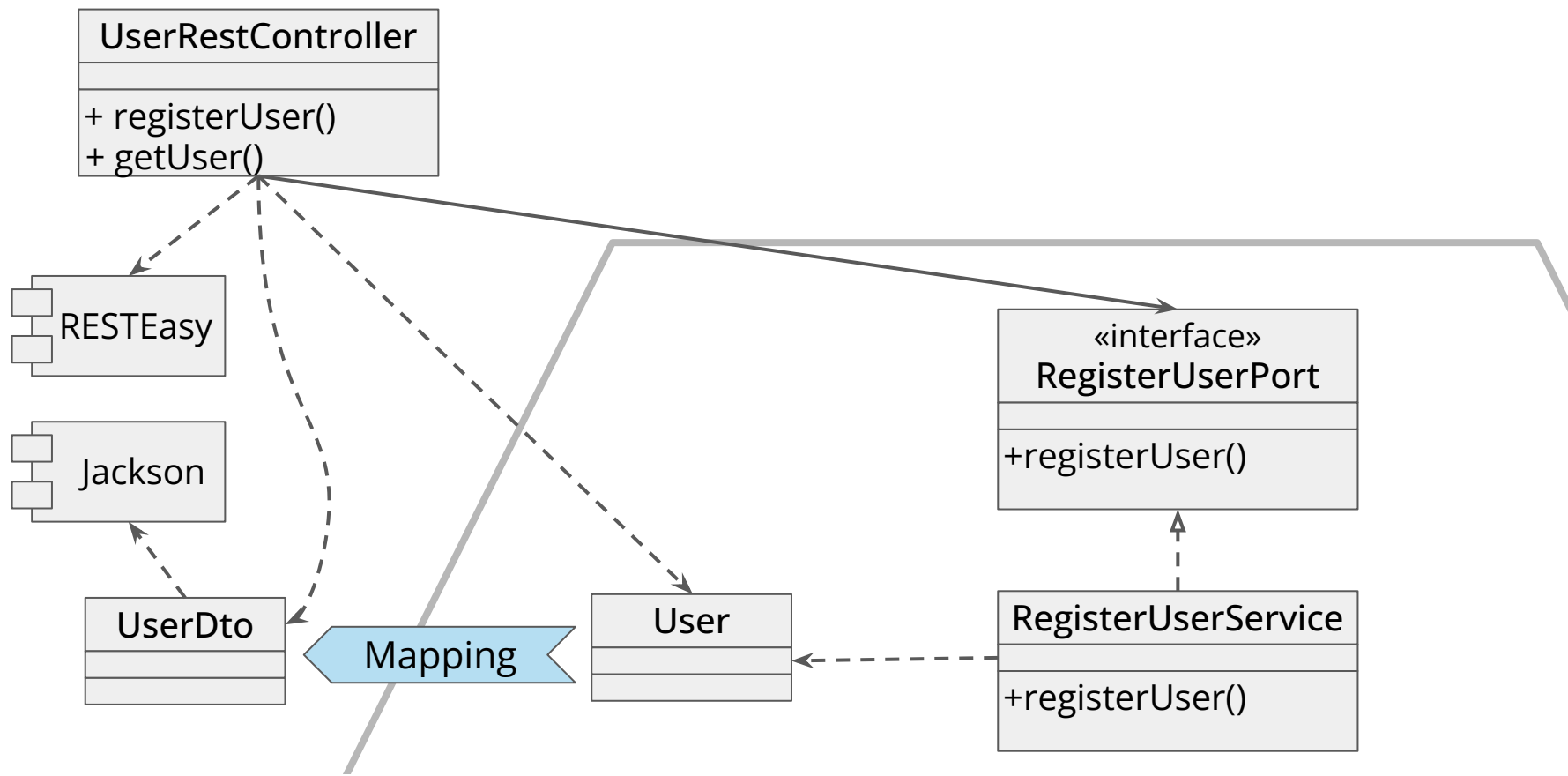
    private Long id;
    private String name;
    private String email;

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

    // ...
}
```

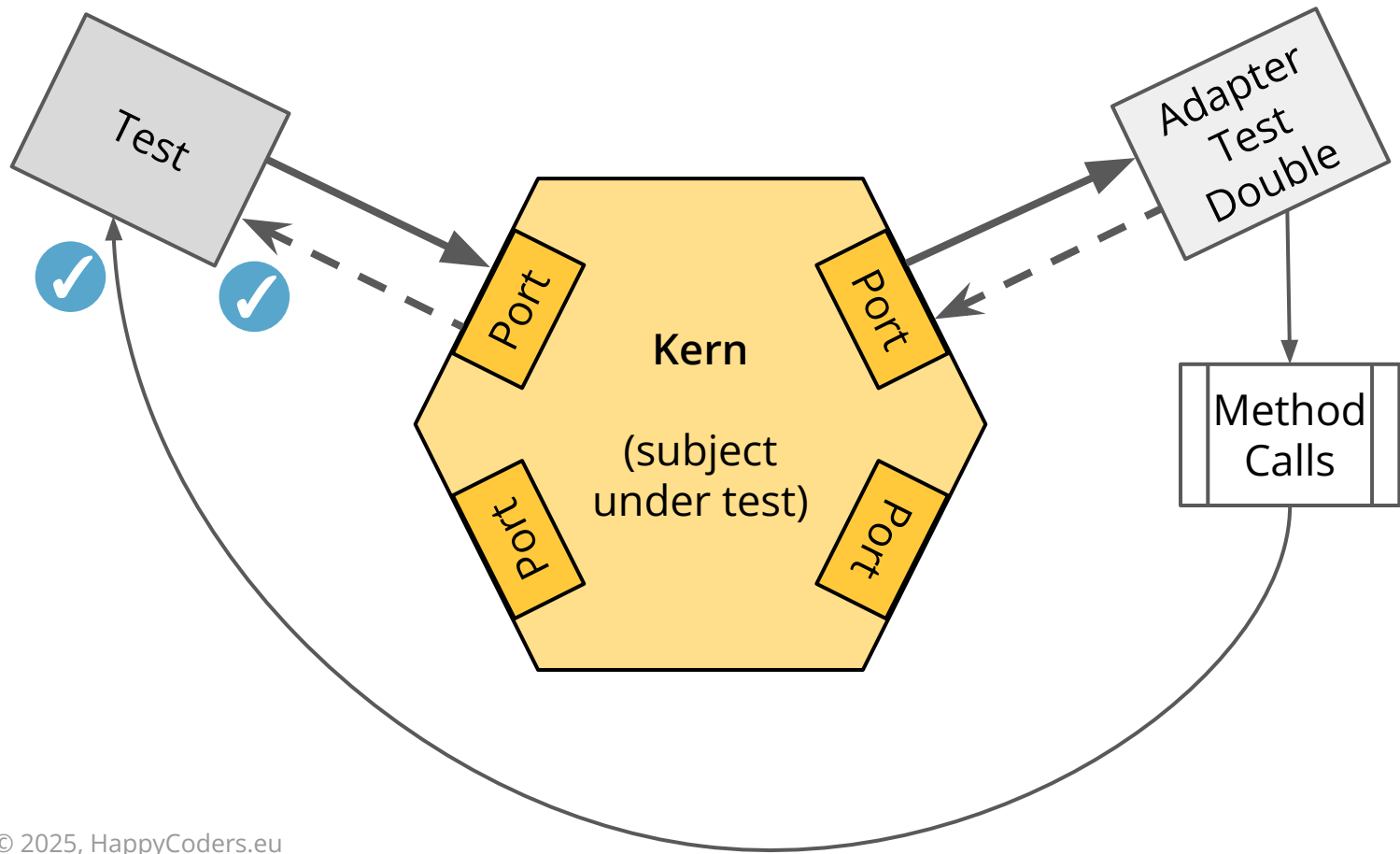


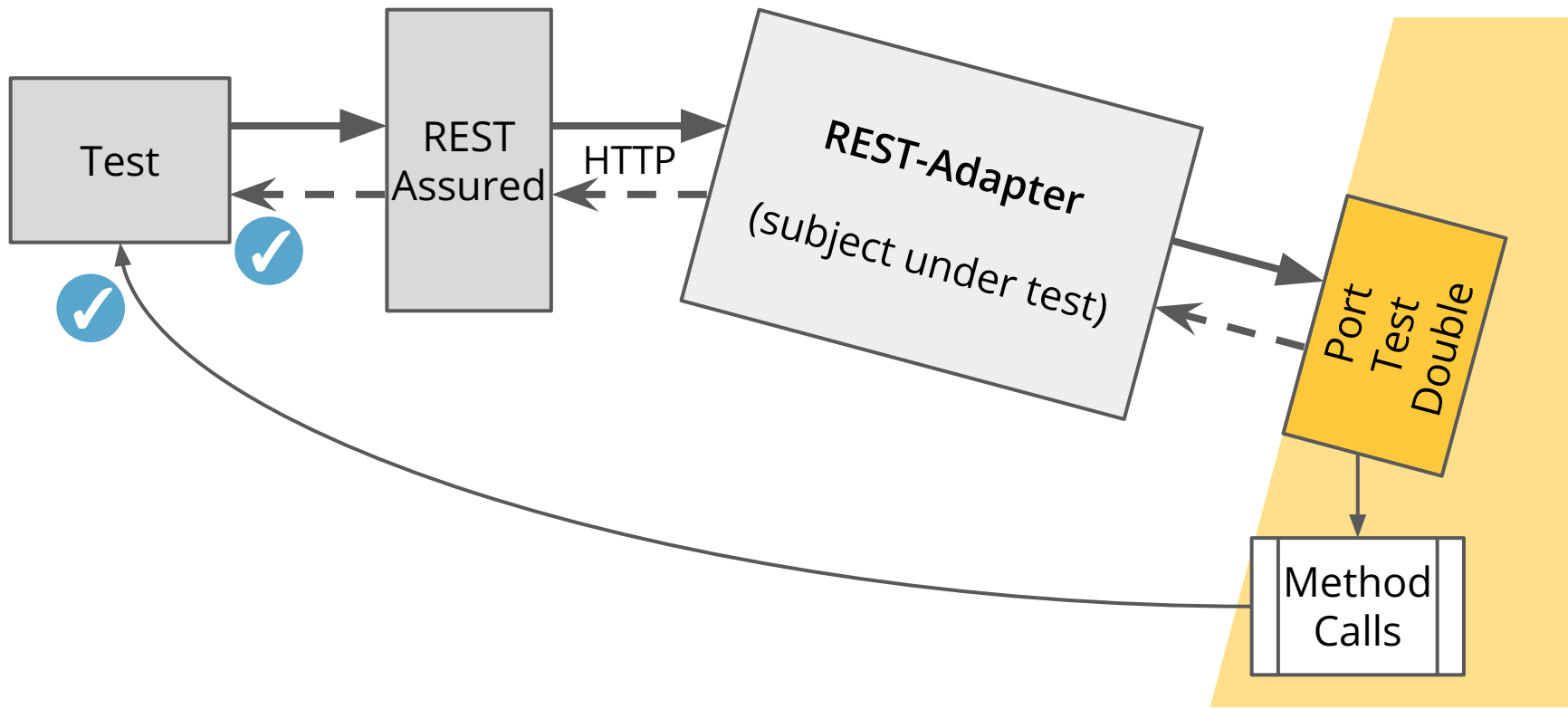


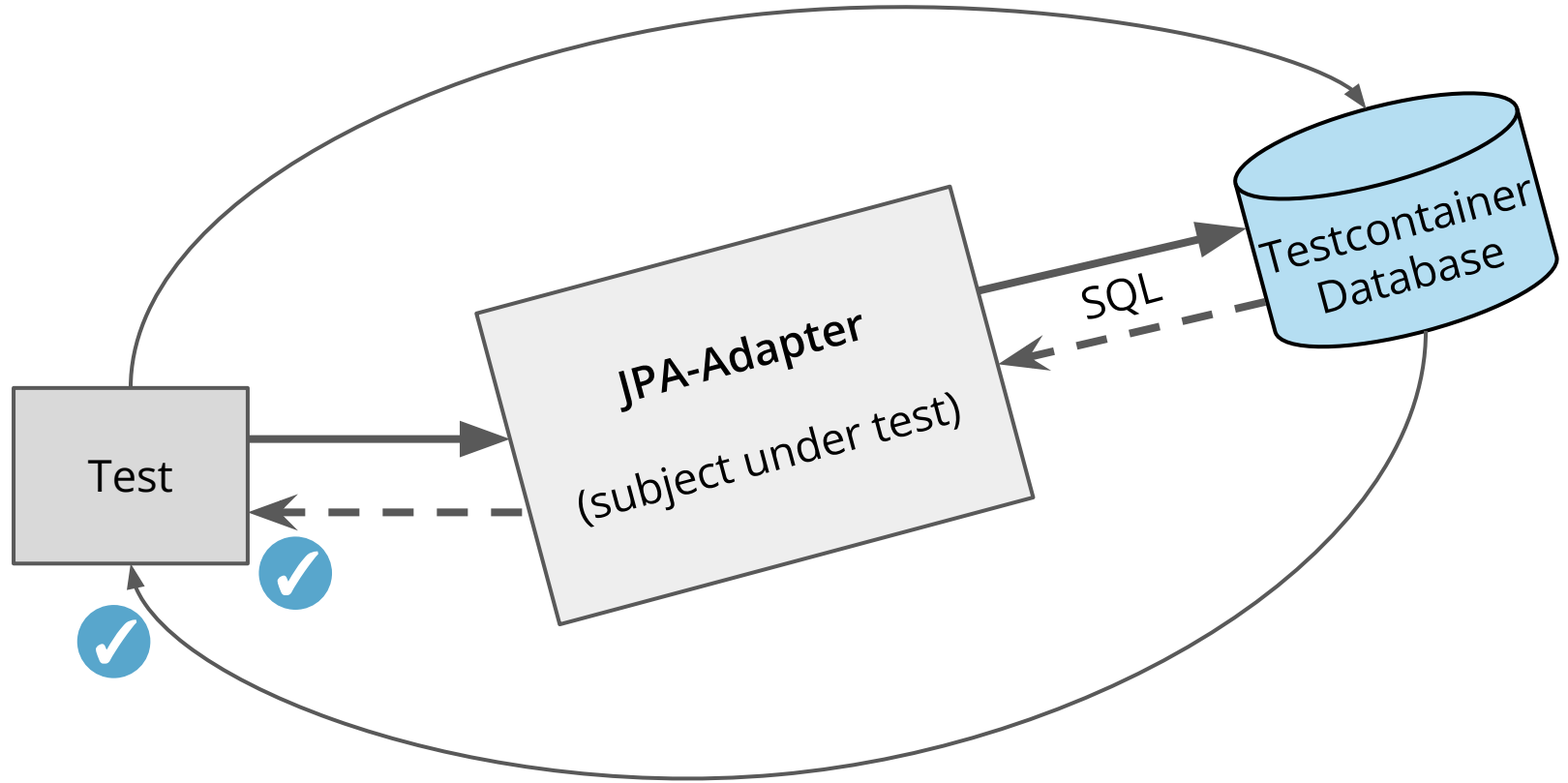


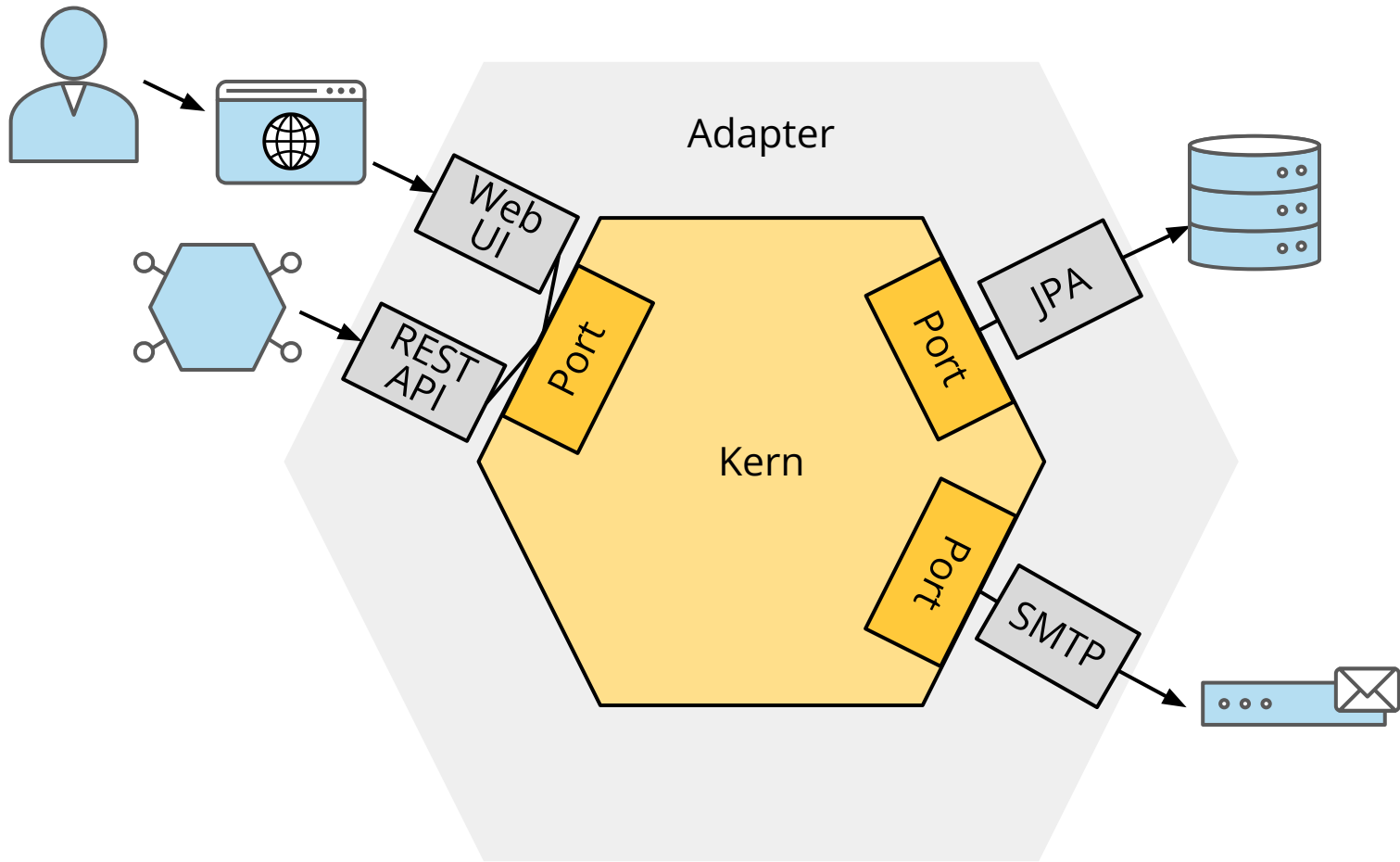
Testbarkeit









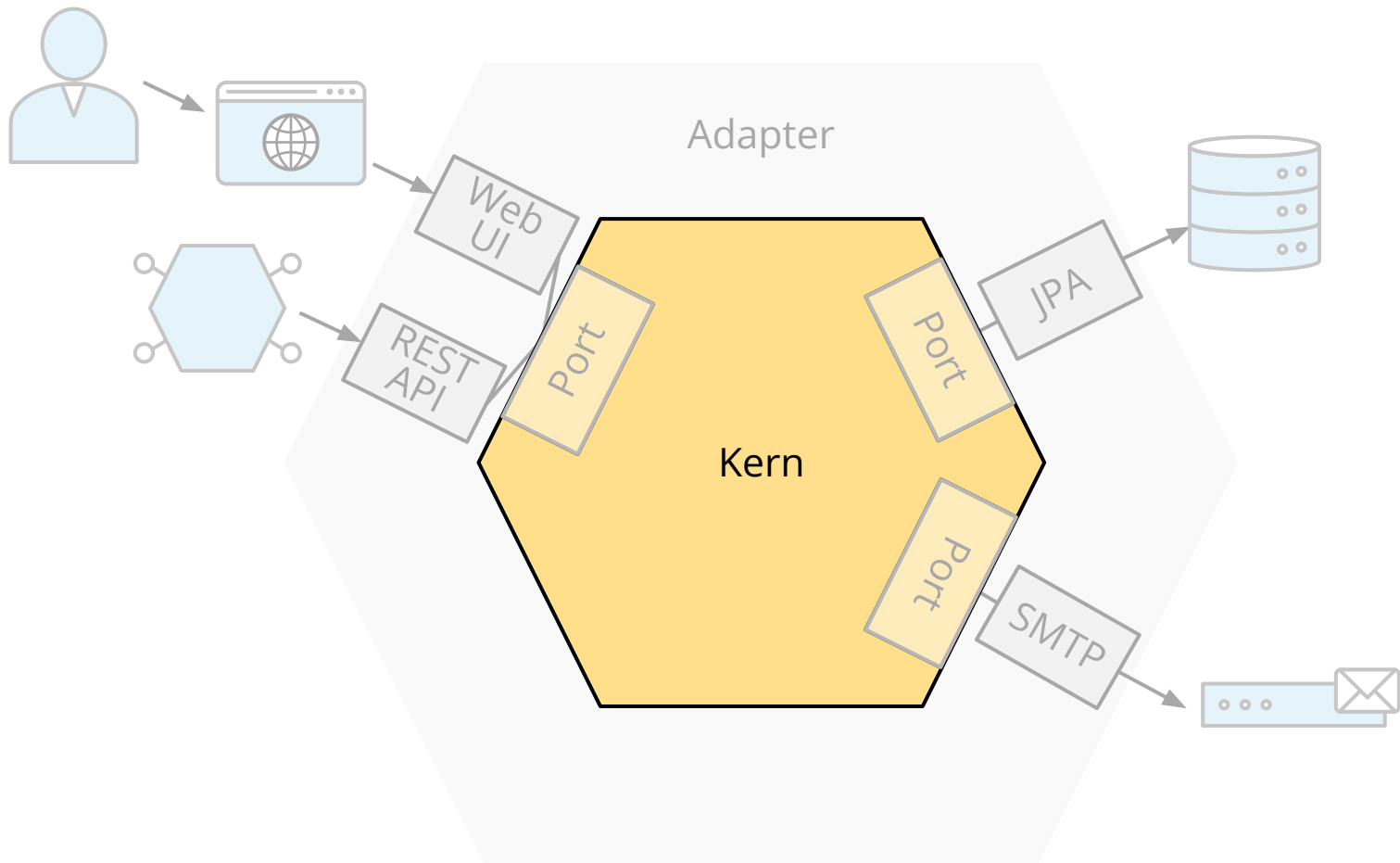


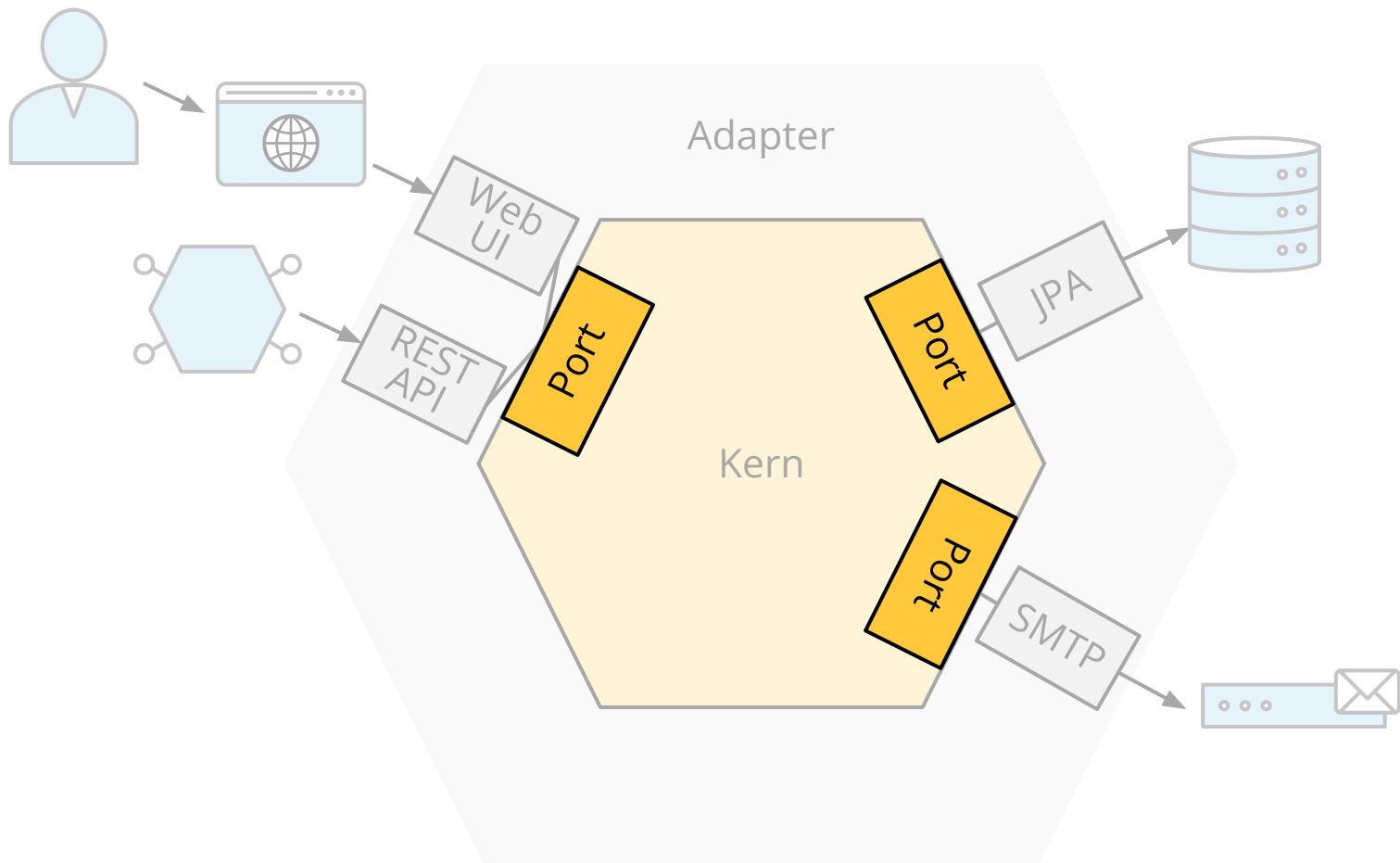
Hexagonale Architektur vs. „Ports & Adapters“

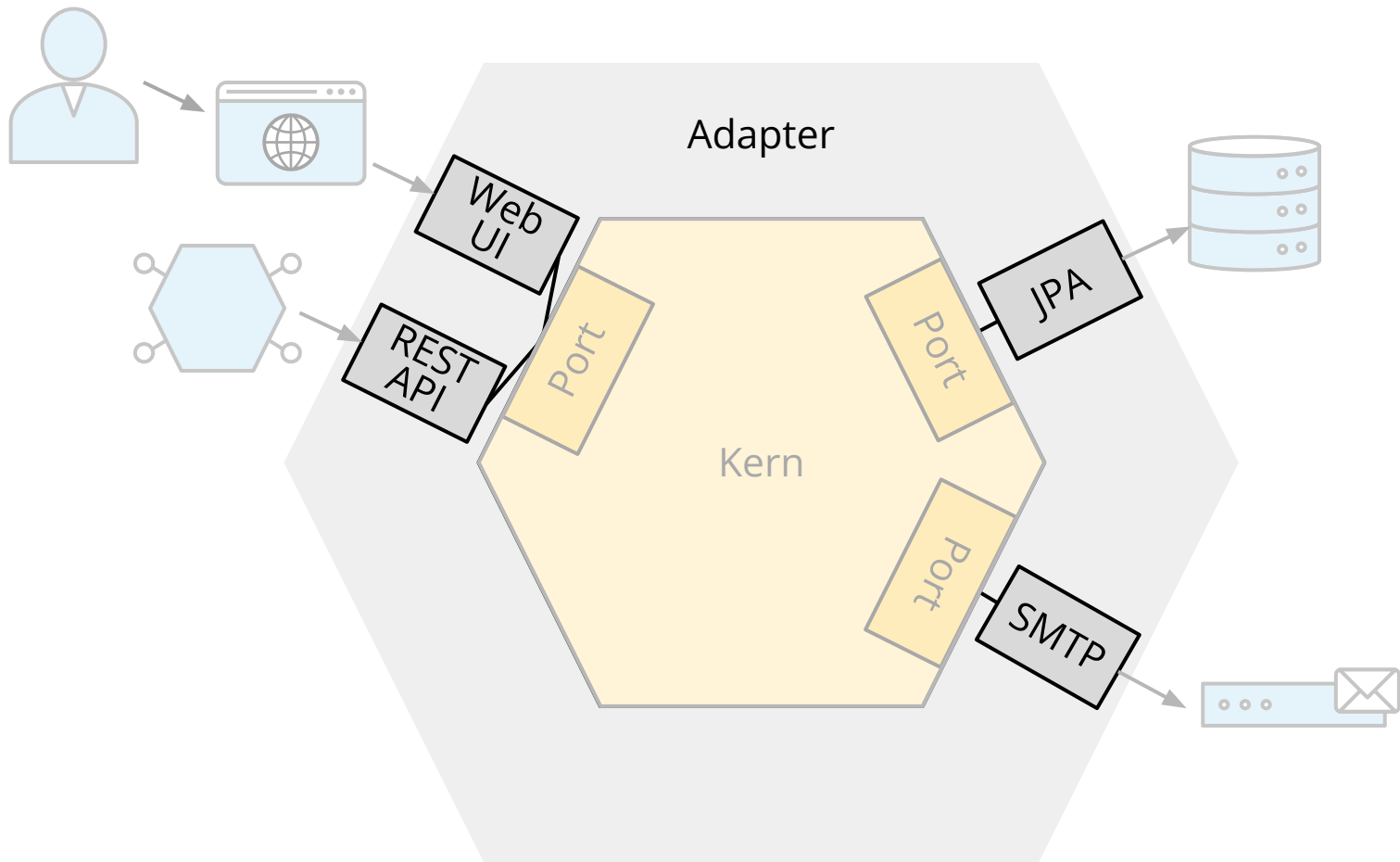


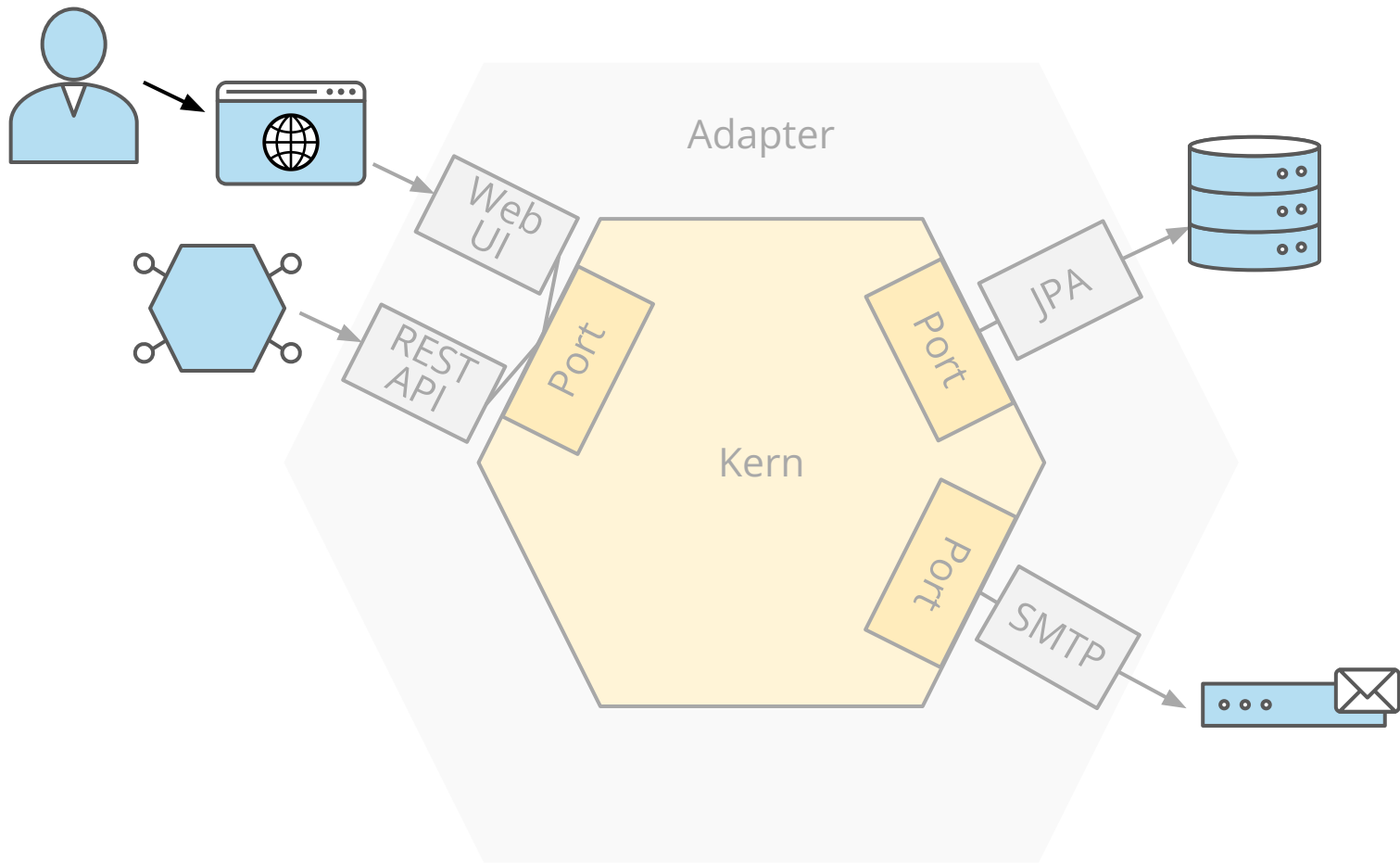
Hexagonale Architektur = „Ports & Adapters“

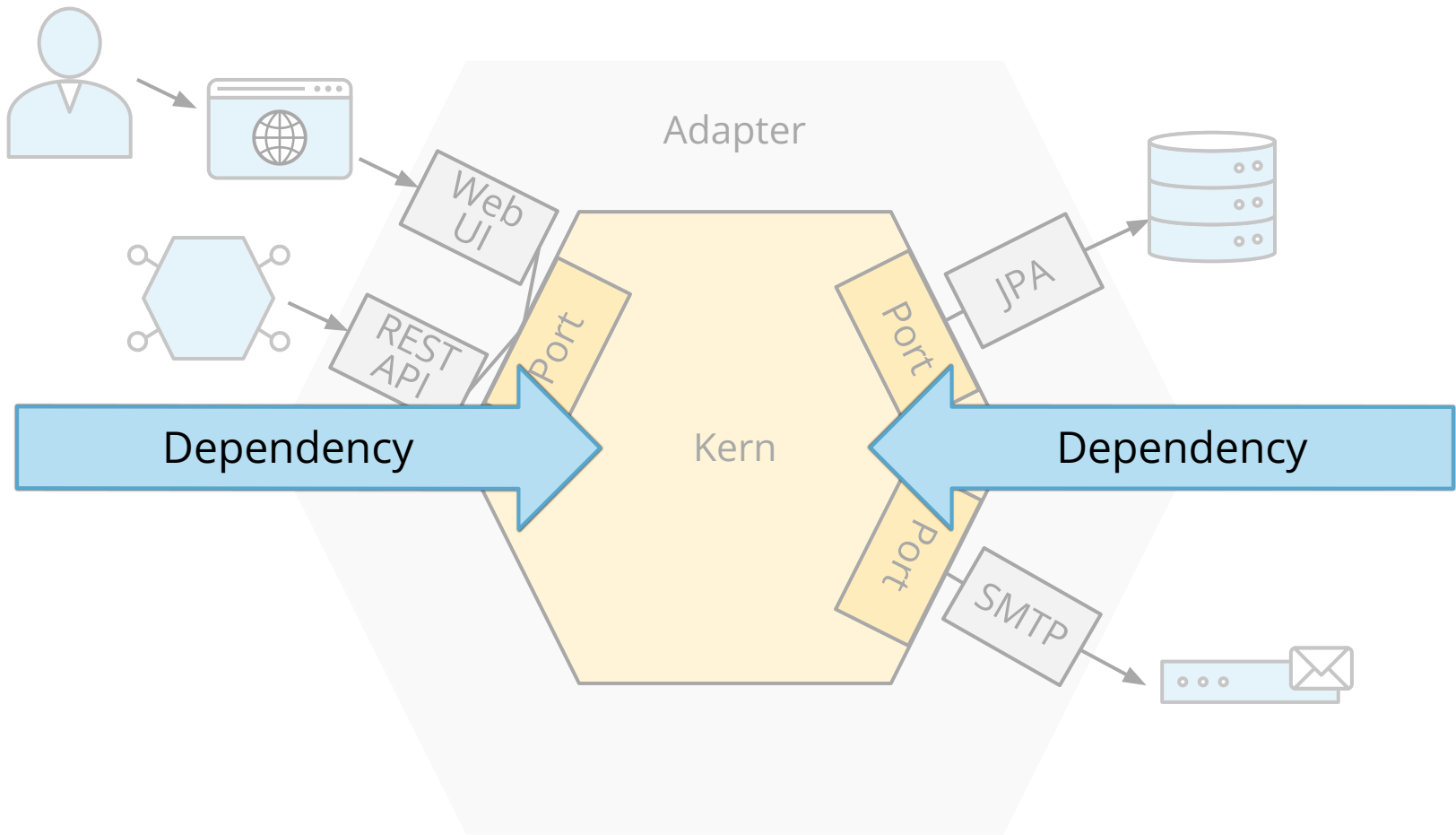












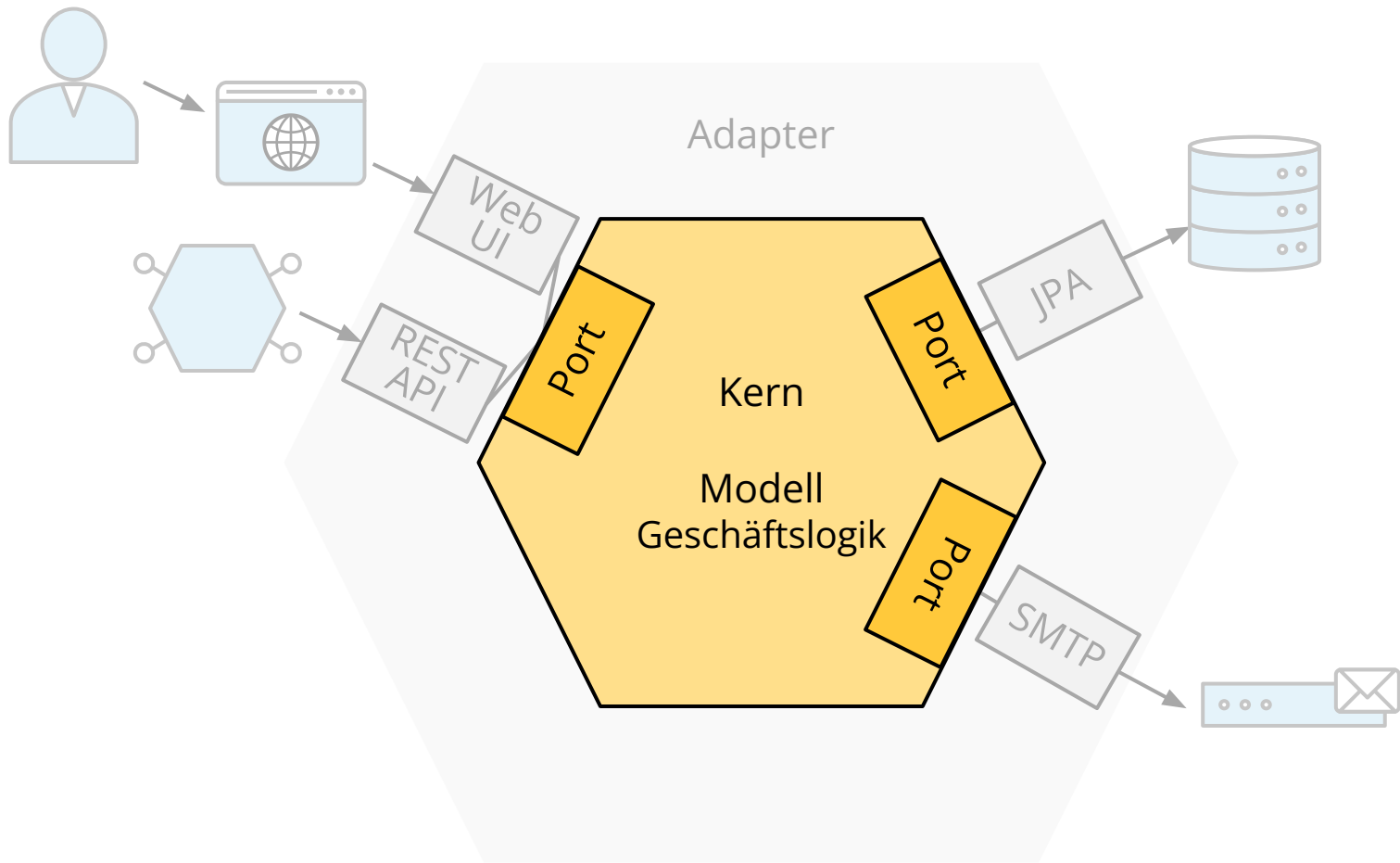
Was ist das Ziel von Softwarearchitektur?

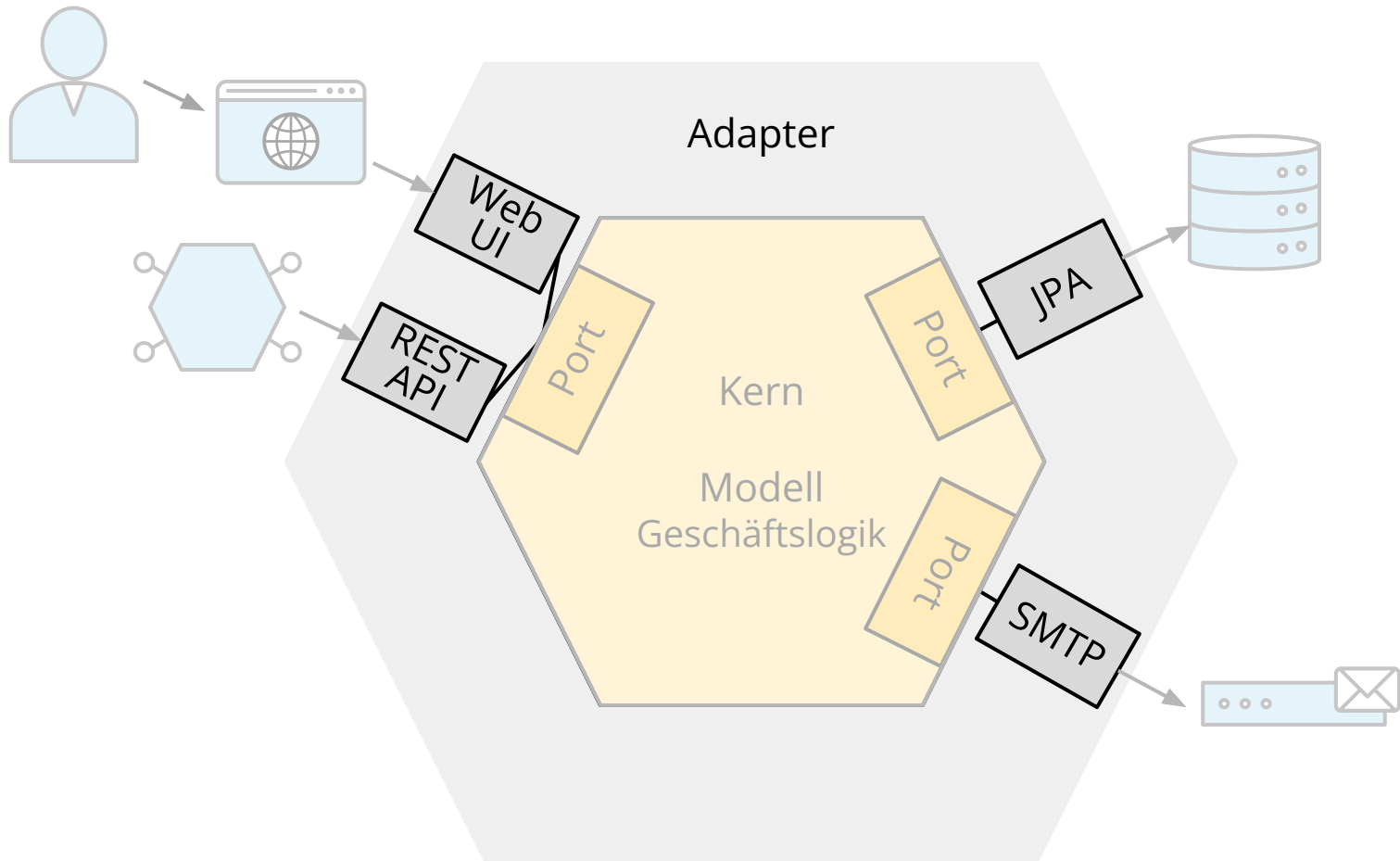
Eine gute Architektur erlaubt es, Software während ihrer Lebensdauer mit möglichst geringem, gleichbleibendem Aufwand zu ändern.

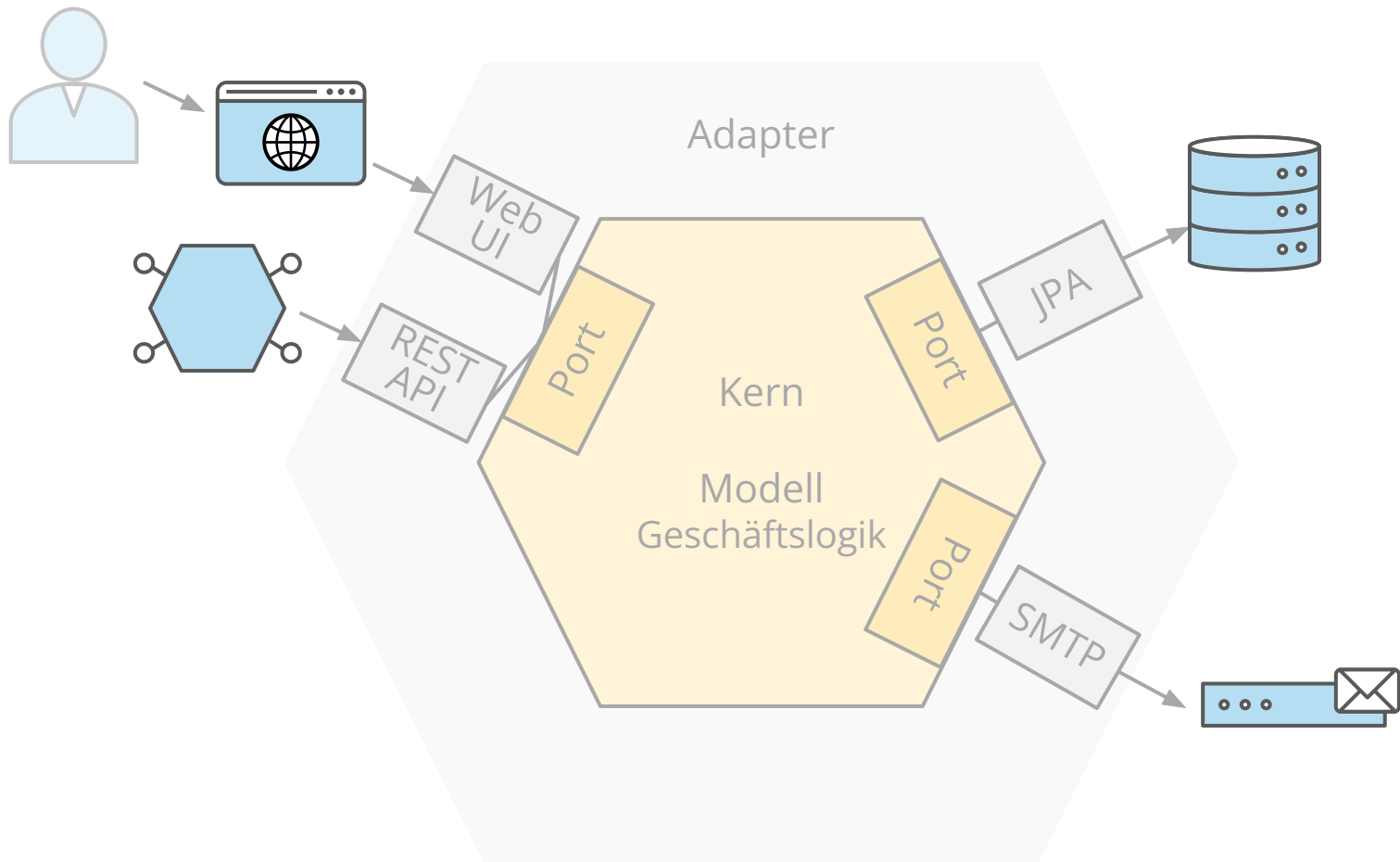


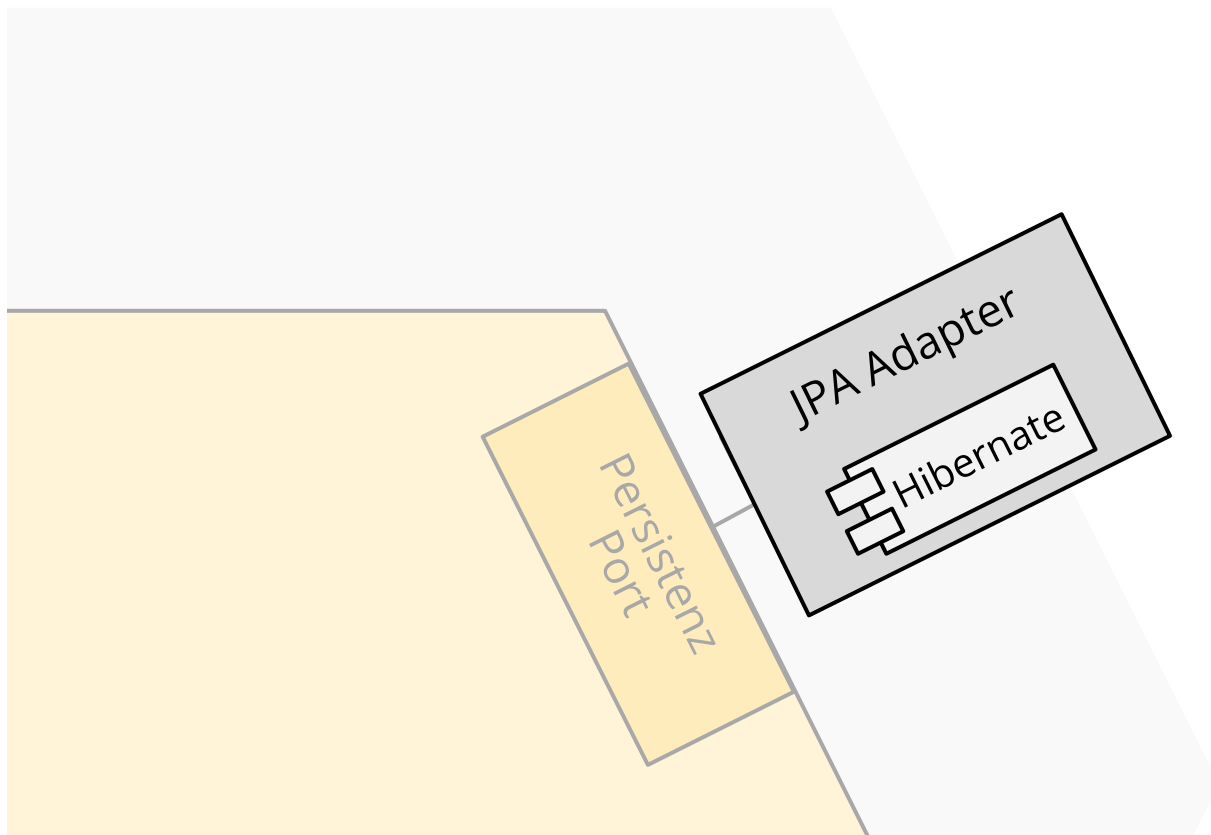
Ziel #1: Änderbarkeit

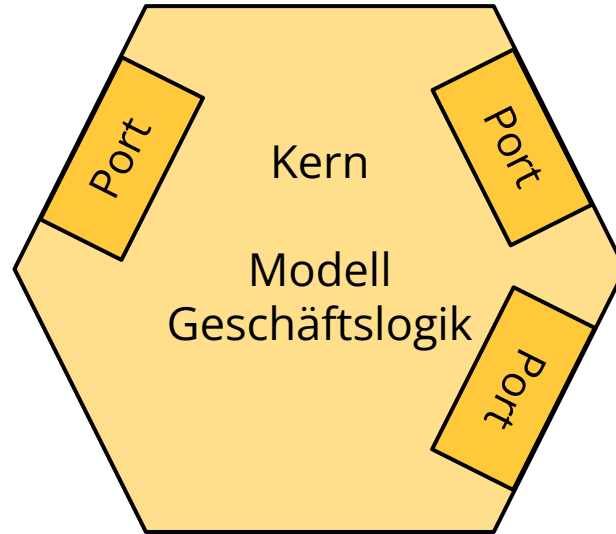






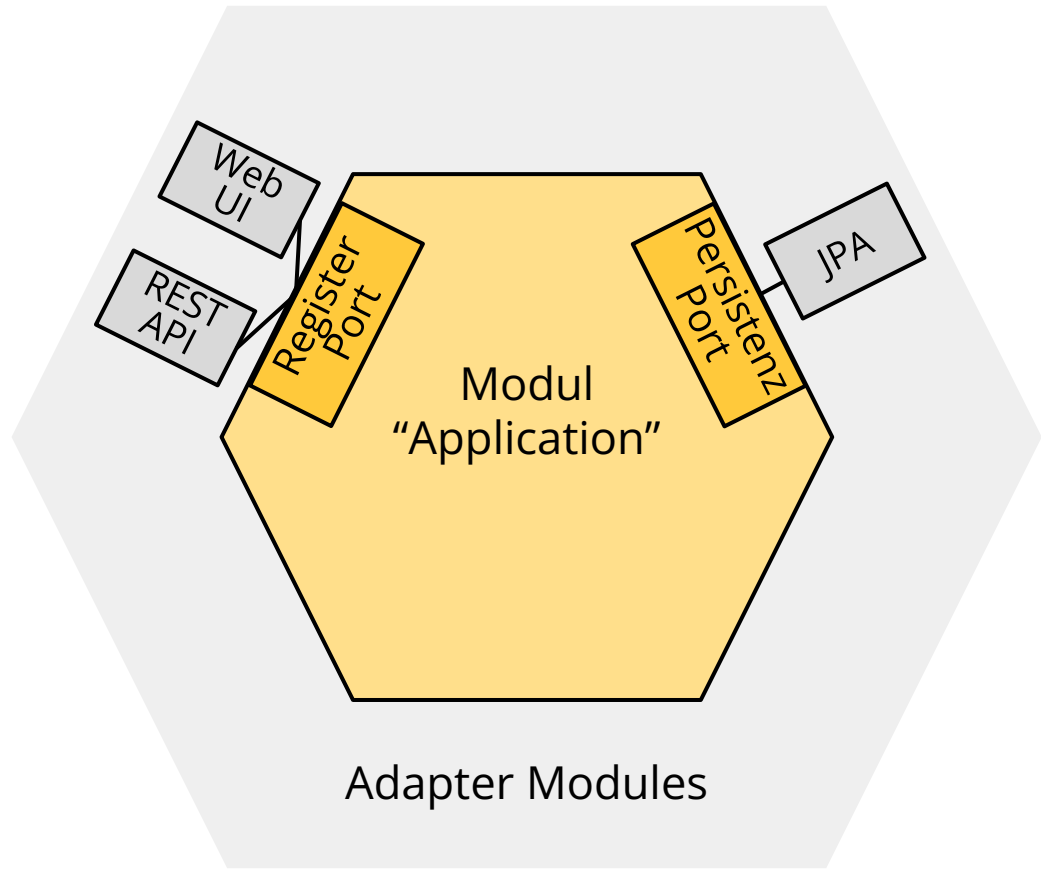






Ziel #2: Lose Kopplung





hexagonal-architecture-java [parent]

> .github

> .idea

adapter-jpa

> src

adapter-jpa.iml

m pom.xml

adapter-rest

> src

adapter-rest.iml

m pom.xml

adapter-web

> src

adapter-web.iml

m pom.xml

application

> src

application.iml

m pom.xml

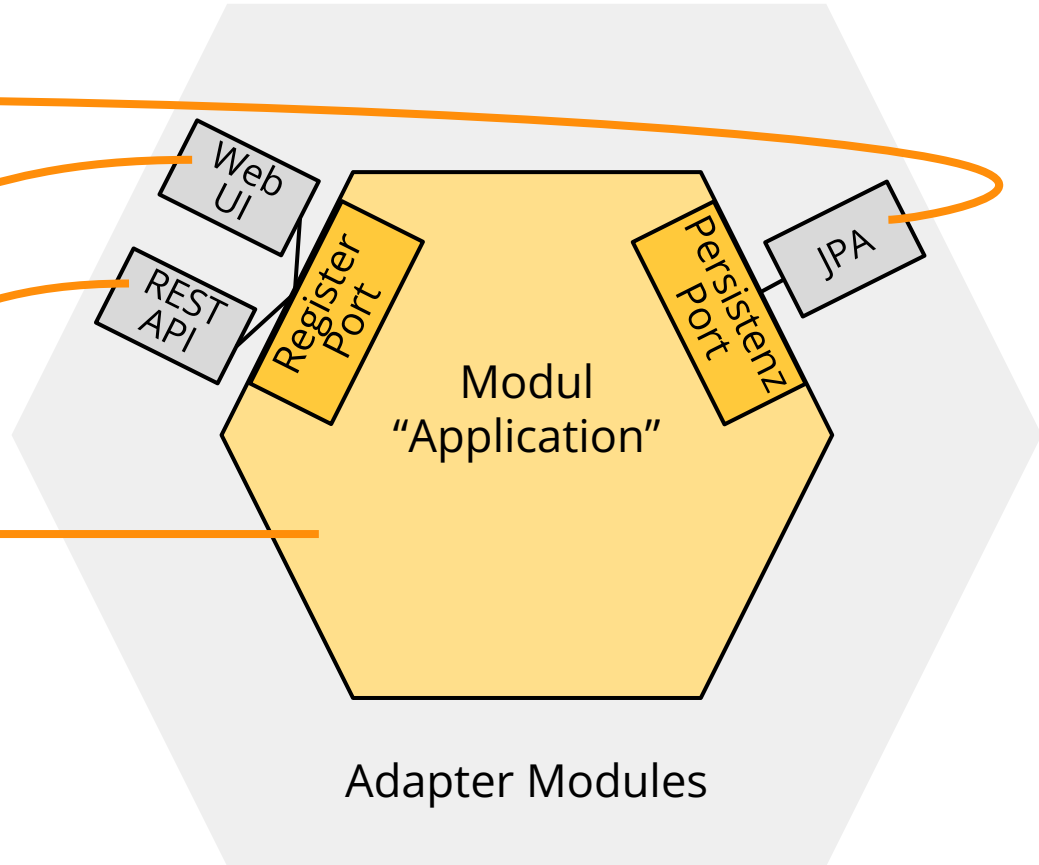
.gitignore

</> google_checks.xml

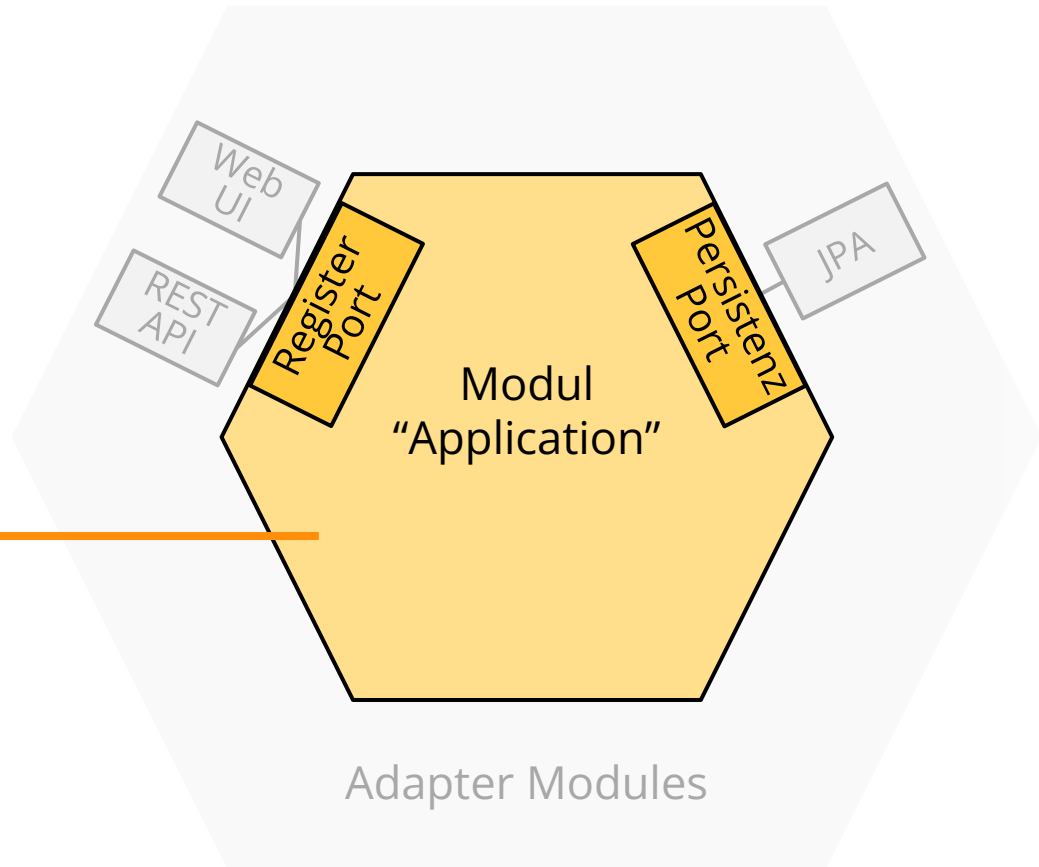
parent.iml

</> pmd-ruleset.xml

m pom.xml



- hexagonal-architecture-java [parent]
 - .github
 - .idea
 - adapter-jpa
 - src
 - adapter-jpa.iml
 - pom.xml
 - adapter-rest
 - src
 - adapter-rest.iml
 - pom.xml
 - adapter-web
 - src
 - adapter-web.iml
 - pom.xml
 - application**
 - src
 - application.iml
 - pom.xml
 - .gitignore
 - google_checks.xml
 - parent.iml
 - pmd-ruleset.xml
 - pom.xml



hexagonal-architecture-java [parent]

> .github

> .idea

adapter-jpa

> src

adapter-jpa.iml

m pom.xml

adapter-rest

> src

adapter-rest.iml

m pom.xml

adapter-web

> src

adapter-web.iml

m pom.xml

application

> src

application.iml

m pom.xml

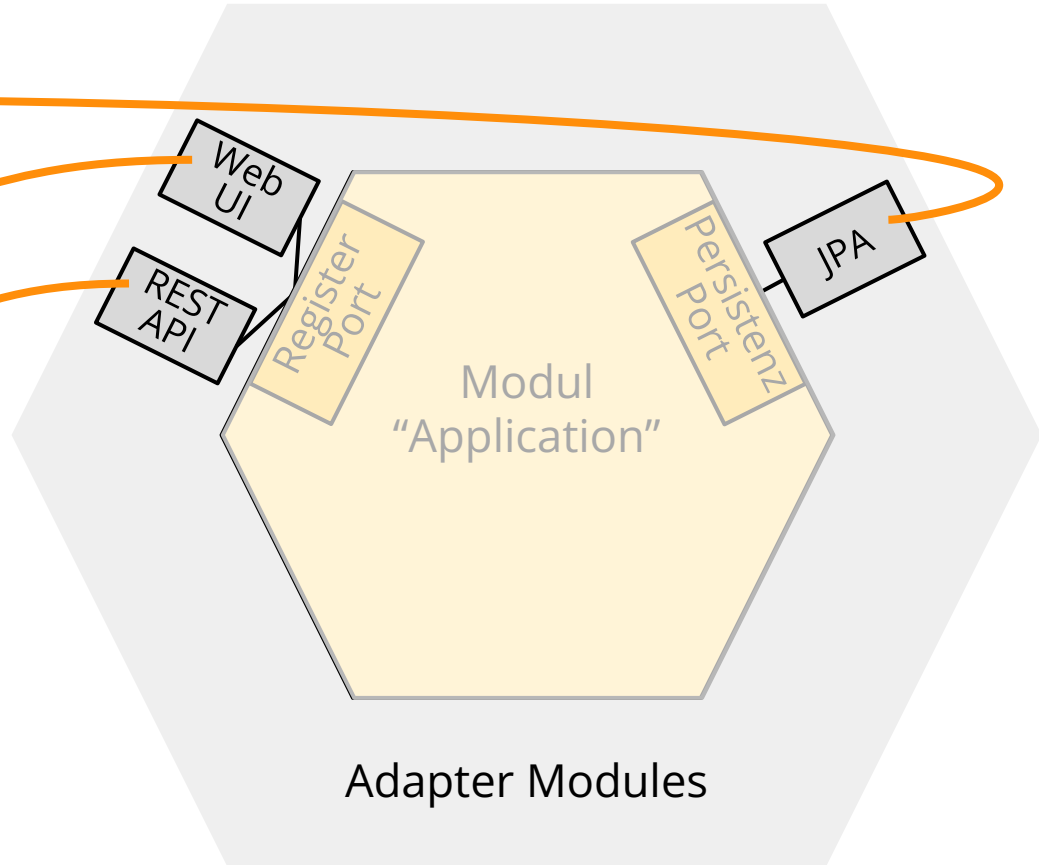
.gitignore

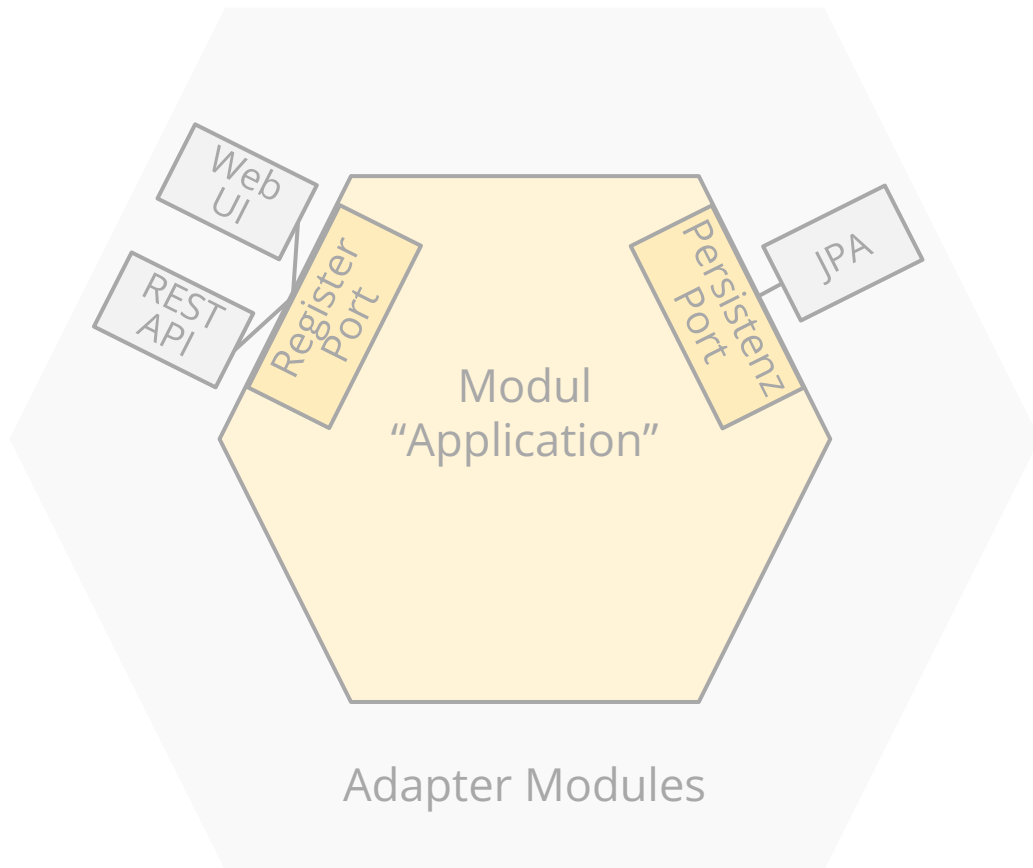
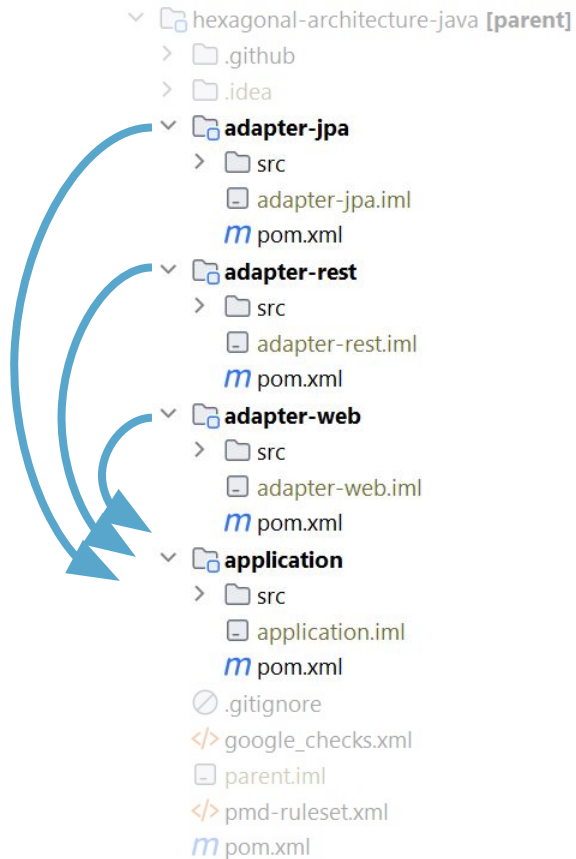
</> google_checks.xml

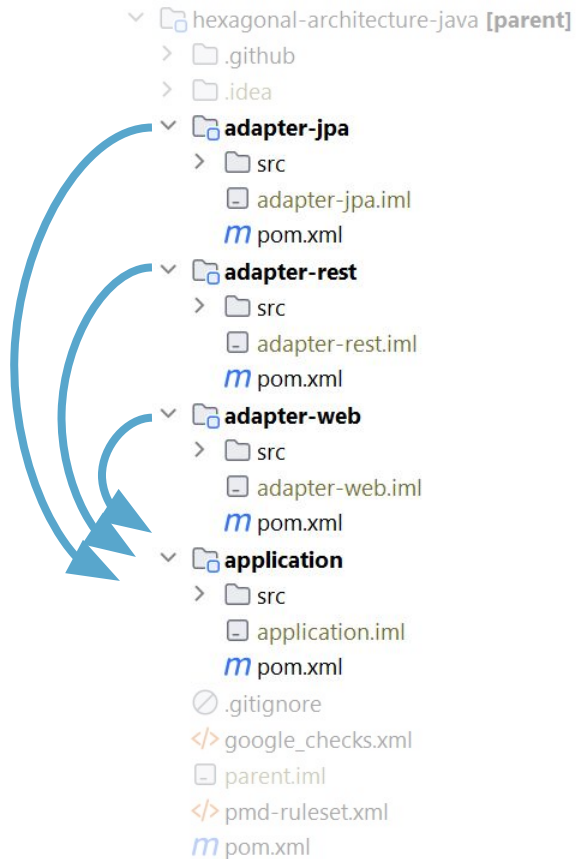
parent.iml

</> pmd-ruleset.xml

m pom.xml





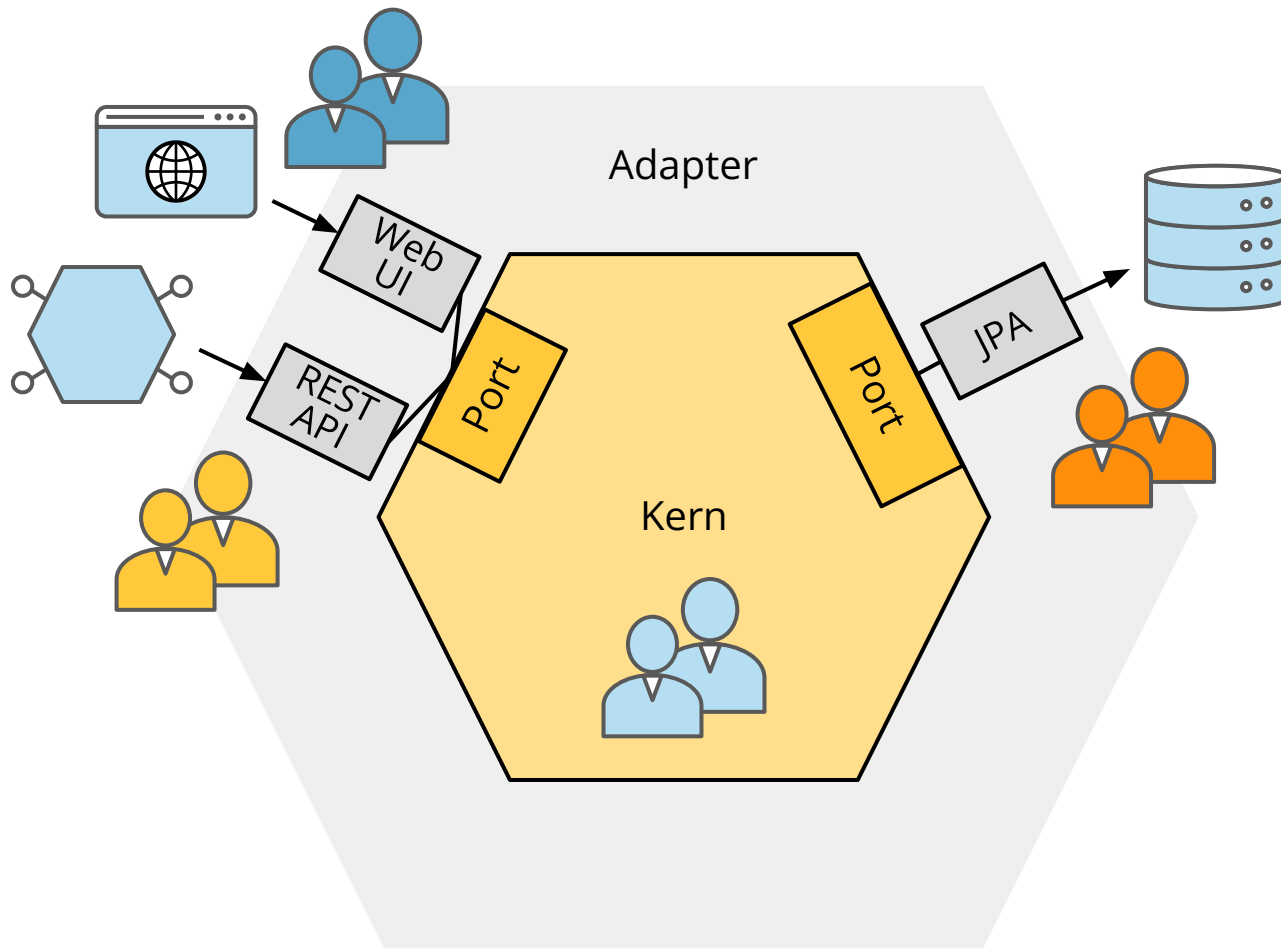


New Java Class	
Ⓒ	Name
Ⓒ	Class
Ⓘ	Interface
Ⓓ	Record
Ⓔ	Enum
@	Annotation



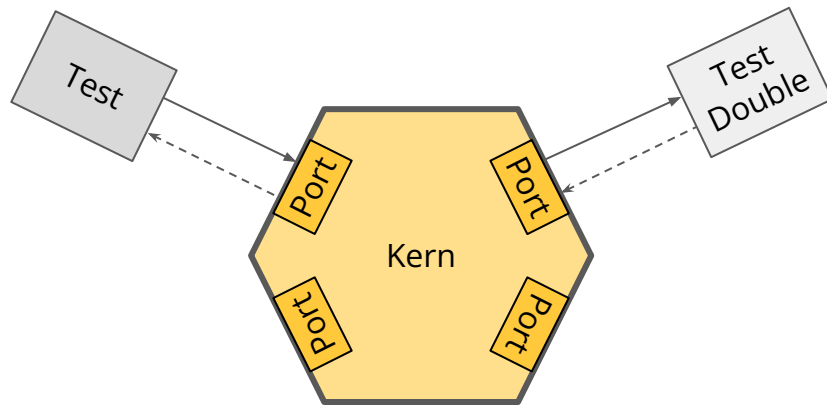
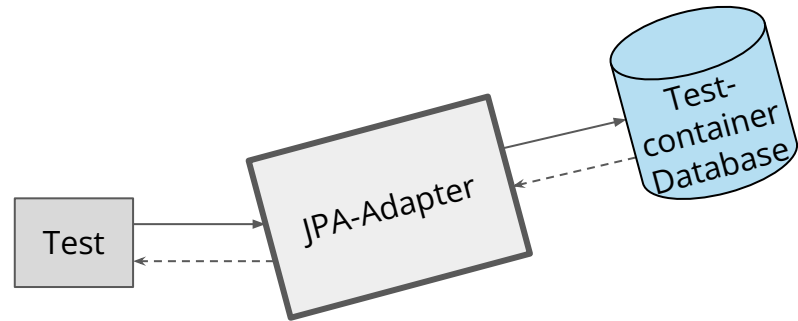
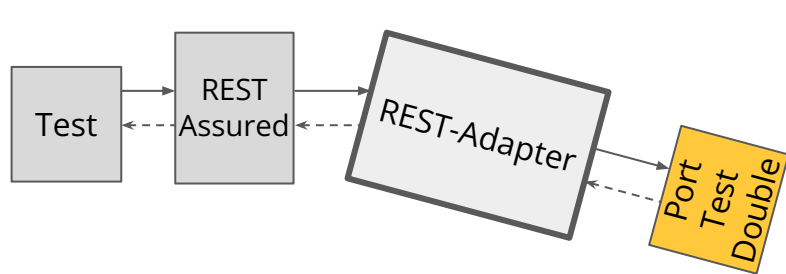
Ziel #3: Unabhängige Entwicklung





Ziel #4: Unabhängige Testbarkeit





Hexagonale Architektur

- ✓ Änderbarkeit
- ✓ Lose Kopplung
- ✓ Unabhängige
Entwicklung
- ✓ Unabhängige Testbarkeit



hexagonal-architecture-java [parent]

> .github

> .idea

adapter-jpa

src

main

java

eu.happycoders.shop.adapter.out.persistence.jpa

test

adapter-jpa.iml

pom.xml

adapter-rest

src

main

java

eu.happycoders.shop.adapter.in.rest

test

adapter-rest.iml

pom.xml

adapter-web

src

main

java

eu.happycoders.shop.adapter.in.web

test

adapter-web.iml

pom.xml

application

src

main

java

eu.happycoders.shop.application

port

in

rest

web

out.persistence

service

test

application.iml

pom.xml

bootstrap

src

main

java

eu.happycoders.shop.bootstrap

resources

test

bootstrap.iml

pom.xml

model

src

main

java

eu.happycoders.shop.model

test

model.iml

pom.xml

doc

.gitignore

google_checks.xml

parent.iml

pmd-ruleset.xml

pom.xml

README.md

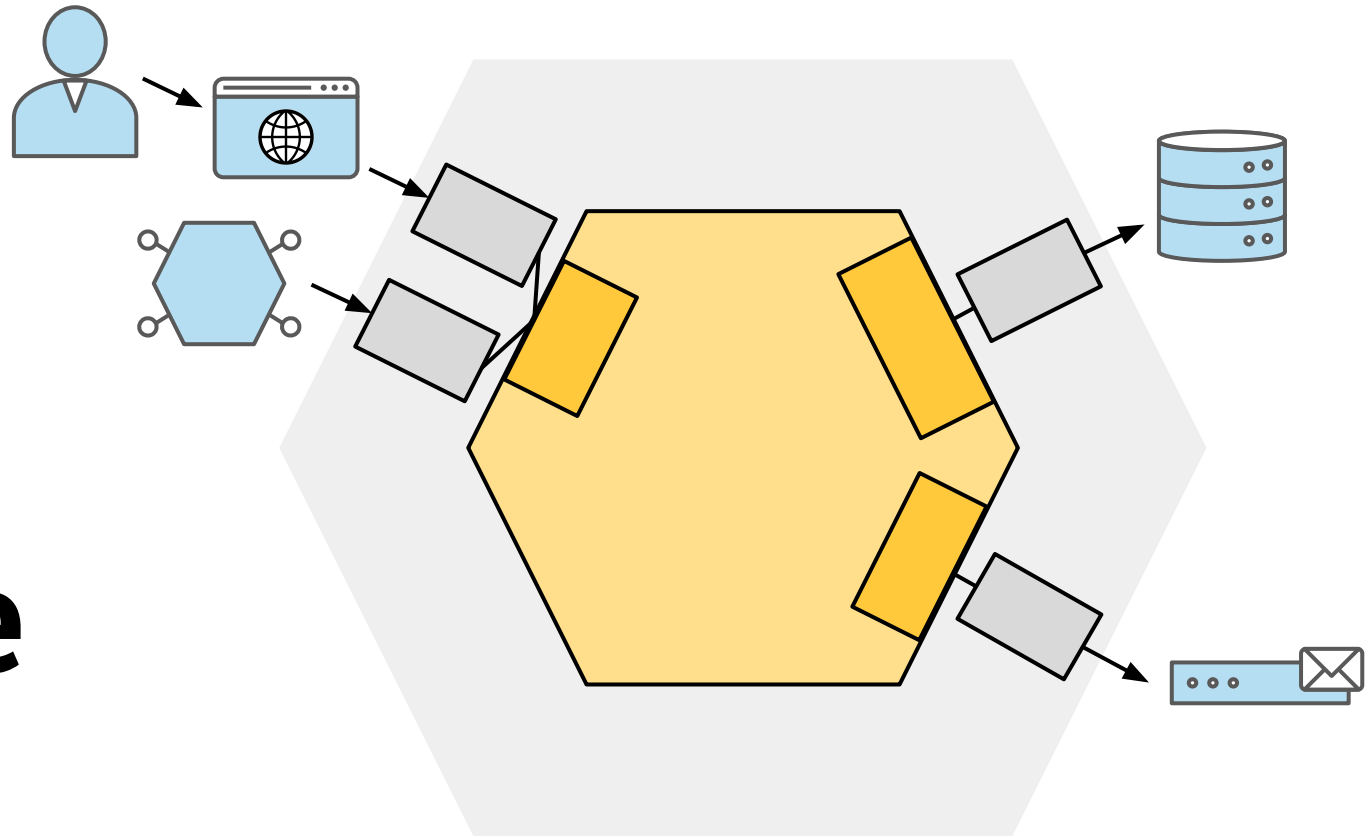
spotbugs-exclude.xml

External Libraries

Scratches and Consoles



Ende



Links



sven@happycoders.eu



<https://www.happycoders.eu/>



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



<https://x.com/svenwoltmann>



<https://bsky.app/profile/sven.woltmann>

