Reaktive Microservices am Beispiel von Lagom

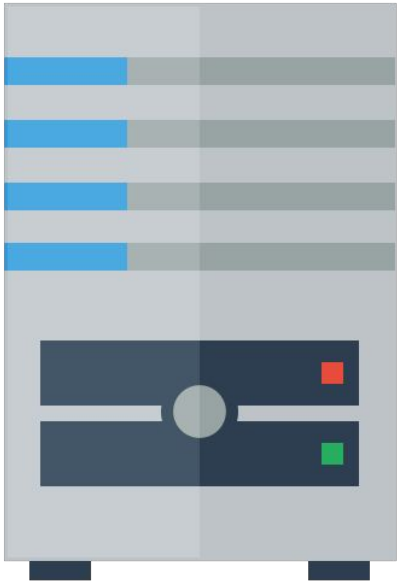
Tim Essig - SS17
Betreuer: Prof. Dr. Zirpins

Microservices

Reaktive Systeme

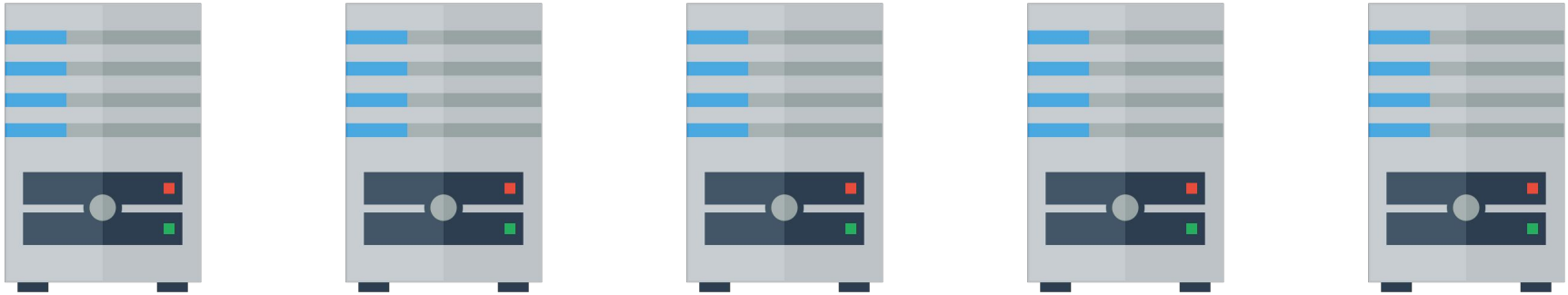
Reaktive Programmierung

Vertikale Skalierung



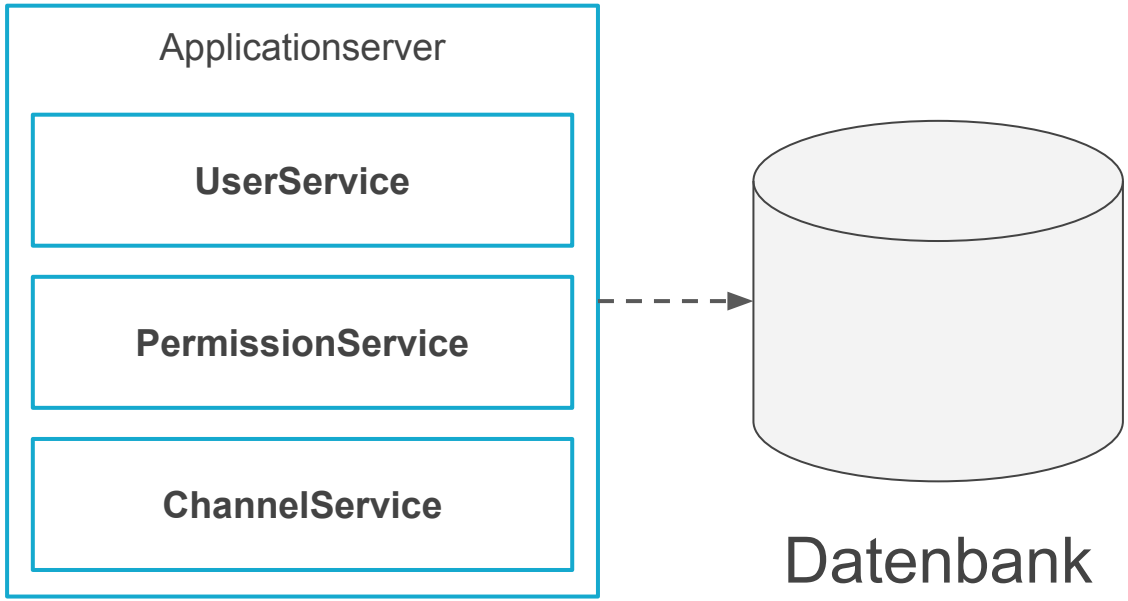
Amdahl’s Law vs. Moore's Law

Horizontale Skalierung

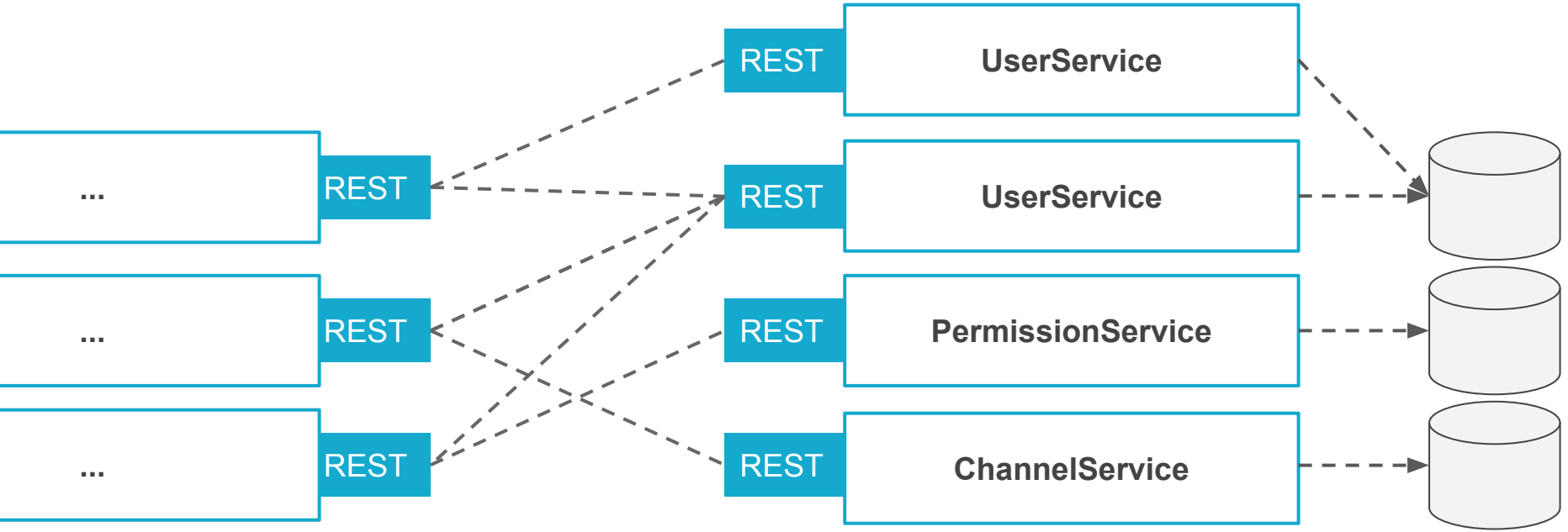


Microservices

Monolith

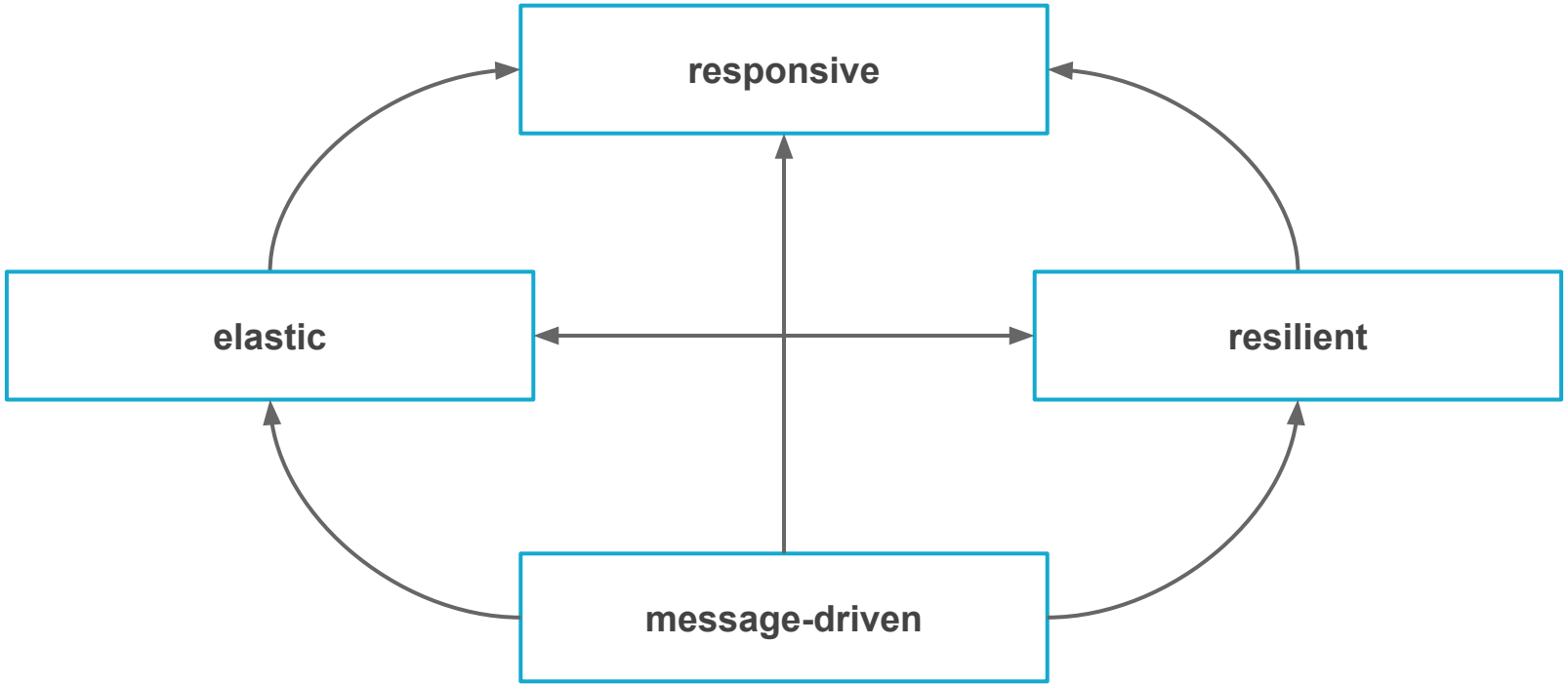

Applicationserver
UserService
PermissionService
ChannelService
Datenbank

Monolith vs. Microservice



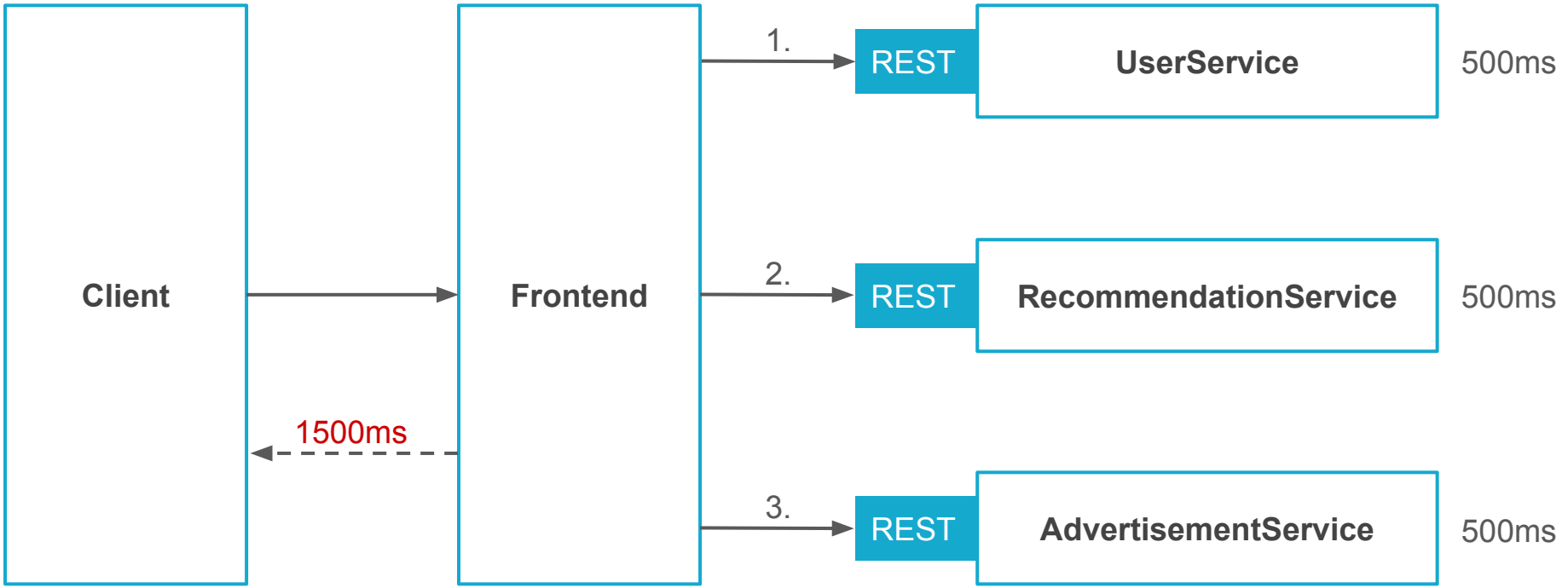
Reaktive Systeme

Reactive Manifesto 2.0

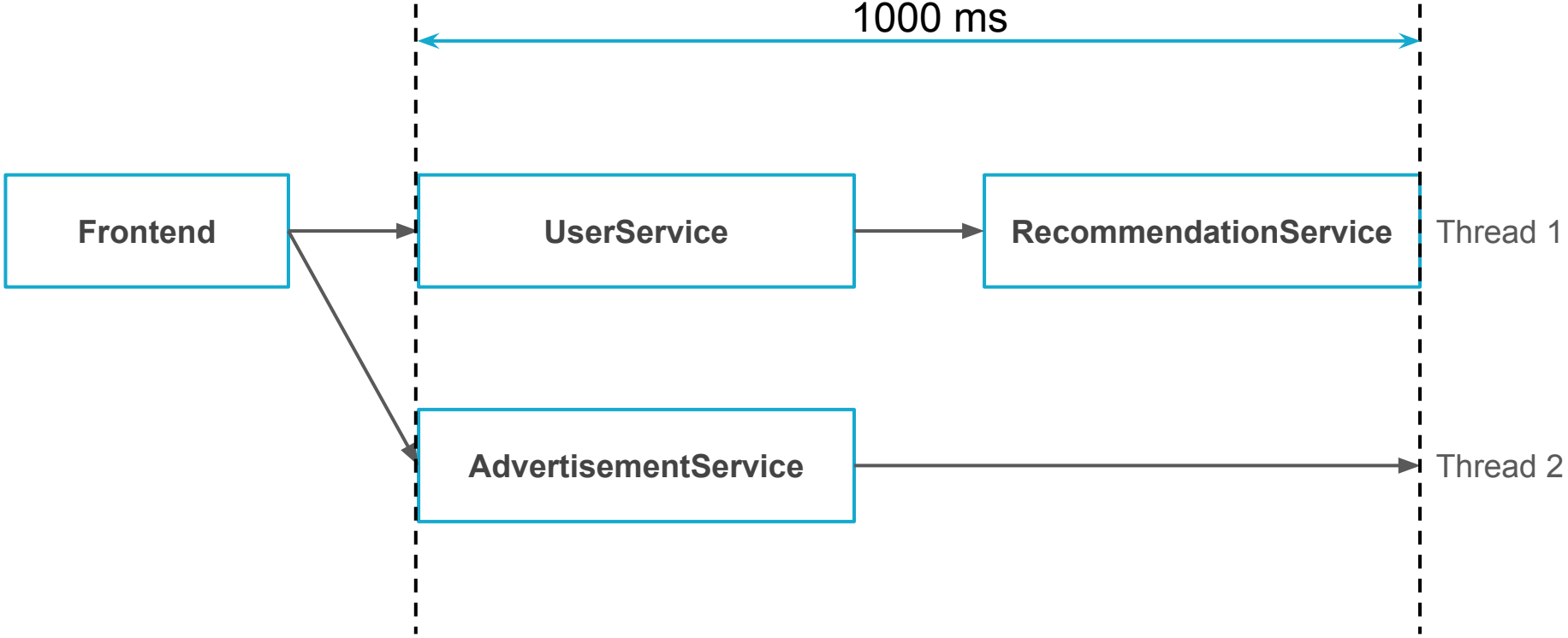


Reaktive Programmierung

Beispielanwendung



Parallelisierung



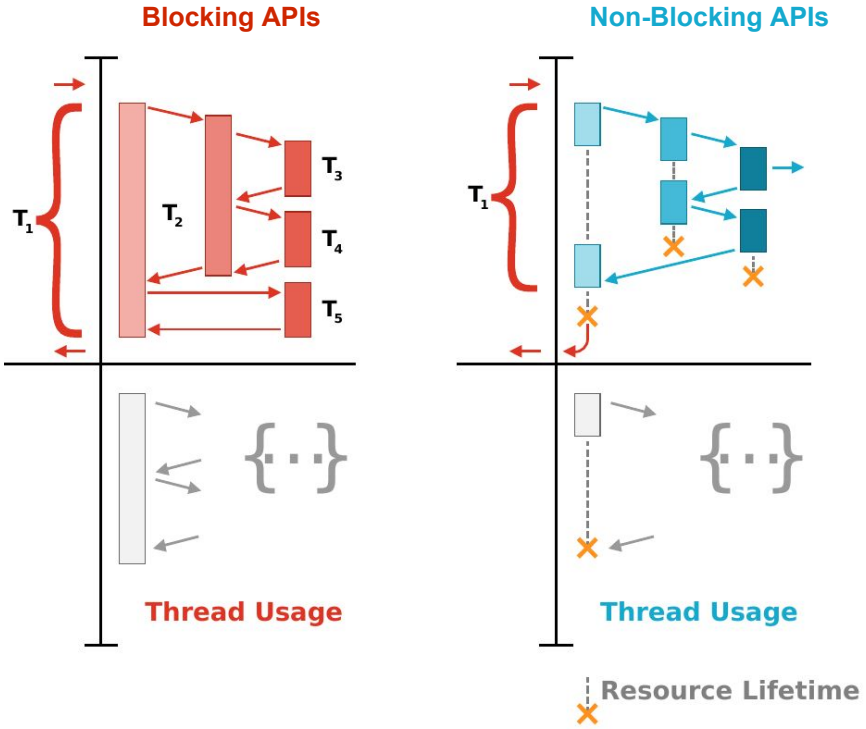
Asynchron vs. Non-Blocking

Future<T>

```
Future<User> future = executor.submit(() -> users.getUser(username) );

User result = future.get(); //Blockiert 500ms
```

Thread Usage

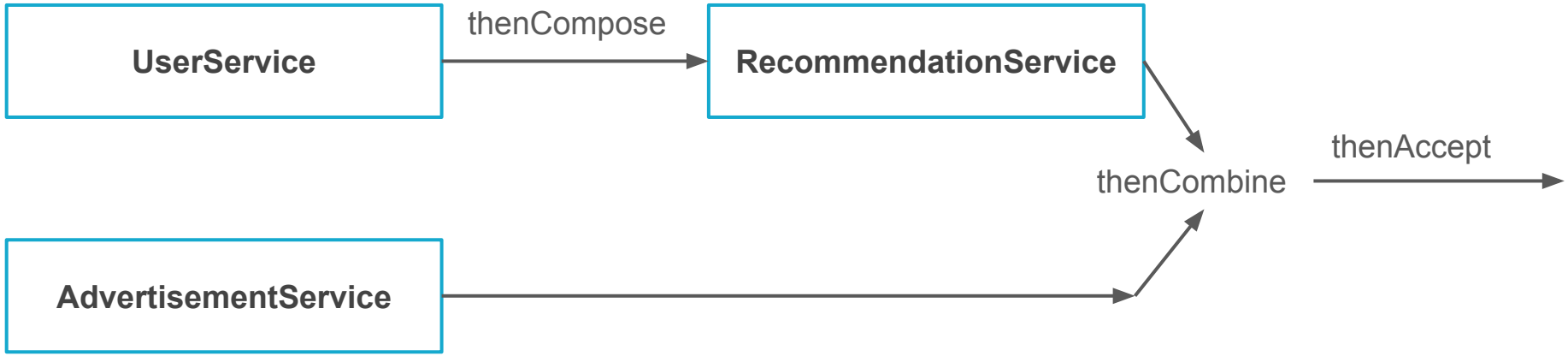


CompletableFuture<T>

Methoden:

- thenCompose - Verkettet zwei CompletableFuture
- thenCombine - Führt zwei CompletableFuture zusammen
- thenAccept - Nimmt das Ergebnis entgegen

CompletableFuture<T>



Demo

Fazit und Diskussion

Blockierende Implementierung

```
1   @GET
2   @Path("/frontend/{user}")
3   public String frontend(@PathParam("user") String userName) {
4         User user = users.lookupUser(userName);
5
7         Recommendation p =
8               recommendations.getRecommendations(user.getUserId());
9
10        Advertisement c = advertisements.getAdvertisement();
11        return template(user, p, c);
12  }
```

Nicht blockierende Implementierung

```
CompletableFuture<Recommendation> cRecommendation =
                                        us.lookupUserCompletable(userName)
 .thenCompose(user -> rs.getRecommendation(user.getUserId()));

CompletableFuture<Advertisement> cAdvertisement =
                            as.lookupAdvertisementCompletable();

 CompletableFuture<Result> cResult =
      cRecommendation.thenCombine(cAdvertisement, (r, a) -> new Result(a, r));

 cResult.thenAccept(result -> asyncResponse.resume( template(result) ));

}
```

Demo