Karlsruher Institut für Technologie

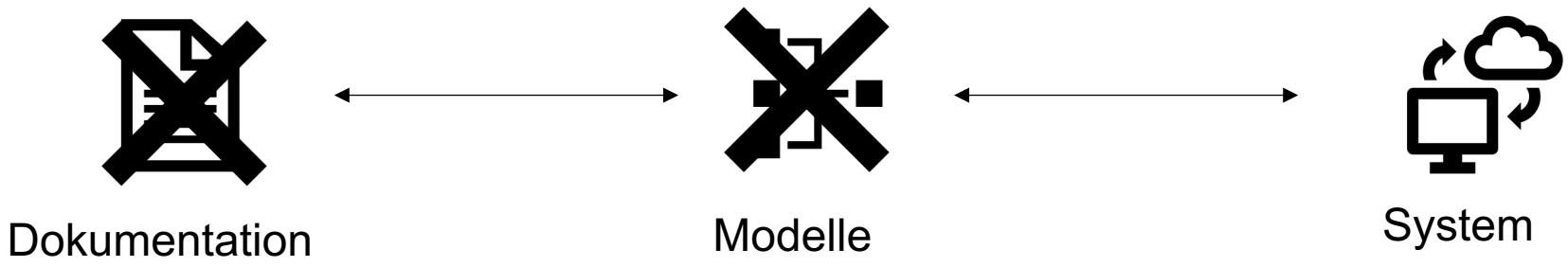
ChatGPT-basierte Extraktion von Architekturmodellen

Hoang Hai Tran, Tim Le Large

Praktikum: Werkzeuge für Agile Modellierung

Motivation

- Arbeit an bestehendem System setzt Dokumentation voraus
- Unvollständige Dokumentation
  - Schweres Verständnis von Legacy Systemen und Open Source Projekten



Motivation

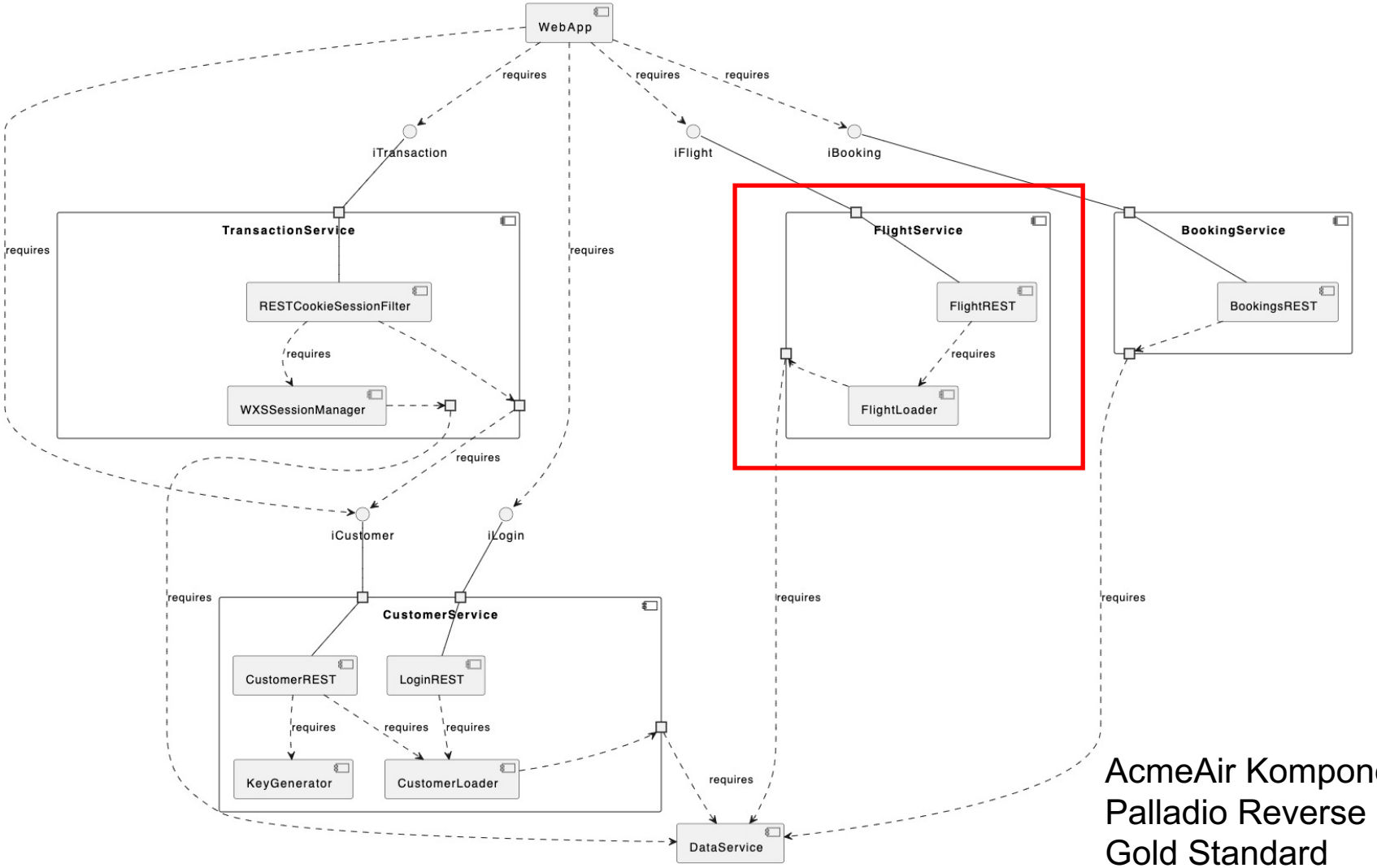
- Lösung: Reverse Engineering von bestehendem Projekt
  - Extraktion von Modellen aus Code

Reverse Engineering

- Viel Arbeit System aus Sourcecode zu modellieren

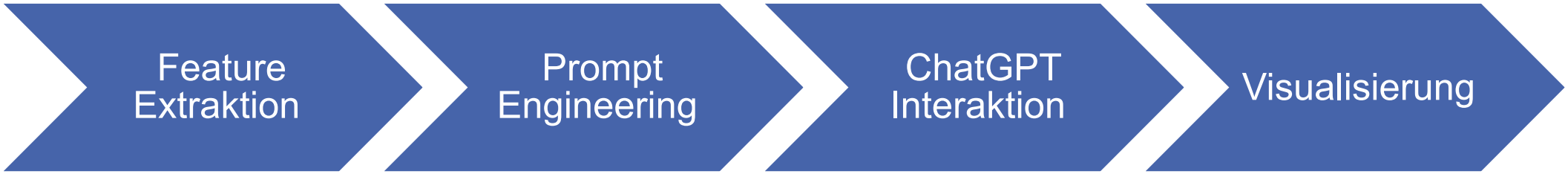
“Lösung” für alles

Modellierung durch ChatGPT



AcmeAir Komponentendiagramm
Palladio Reverse Engineering Benchmark
Gold Standard

Pipeline


Feature Extraktion
Prompt Engineering
ChatGPT Interaktion
Visualisierung

1. Feature Extraktion

- Nutzung des File Retrievers des PalladioSimulators
  - Automatisches finden von Files
    - [JSON, YAML, JAVA, CSV, ECMA, SQL, XML, PROP]
  - Fokus auf **JAVA** Dateien
  - Generierung eines Syntaxbaums des Codes

1. Feature Extraktion

- Imports
- Packages
- Superklassen/ Interfaces
- Modifiers
- Fields
- Methods
- Parameter

1. Feature Extraktion

- Erweiterungen:
  - Anpassungen:
    - Nur **lokale** Imports
    - Nur **public** Fields/Methoden
    - Nur **nicht primitive** Parameter
  - Projektspezifische Features:
    - **Piggymetrics** (SpringBoot) & **ACMEAir** (Plain Java)
    - Z.B. Spring Boot => Annotation
  - Docker Files
    - Weitere Details über Microservices

1. Feature Extraktion

Class: ResourceServerConfig
extends: ResourceServerConfigurerAdapter
Package: package com.piggymetrics.account.config;
imports: Modifiers: @Configuration, @EnableResourceServer, public
Field: sso: ResourceServerProperties private, final
Method: ResourceServerConfig(ResourceServerProperties sso):
Method: clientCredentialsResourceDetails(): ClientCredentialsResourceDetails
Method: oauth2FeignRequestInterceptor(): RequestInterceptor
Method: clientCredentialsRestTemplate(): OAuth2RestTemplate
Method: tokenServices(): ResourceServerTokenServices
Method: configure(HttpSecurity http): void

2. Prompt Engineering

- Source: A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT
- “The quality of the output(s) generated by a conversational LLM is directly related to the quality of the prompts provided by the user.”

- Nutzung von Prompt Pattern für Systematischen Ansatz
  1. Flipped Interaction Pattern
  2. Recipe Pattern
  3. Persona Pattern
  4. Template Pattern

Prompt Engineering – Flipped Interaction Pattern

- Frage ChatGPT, wie ein Komponentendiagramm erstellt werden sollte

"What approach would you suggest for using prompt engineering to generate a component diagram from Java files of a software project?"

Prompt Engineering – Recipe Pattern

- Gib ein Rezept an, nach welchem ChatGPT arbeiten sollte

Steps to Create a Component Diagram

1. Identify Components:
   - Group related classes into logical components.
   - Example: Classes related to customer management can be grouped into a CustomerService component.

2. Define Interfaces:
   - Identify the methods that will serve as interfaces for each component.
   - These are the public methods that other components will interact with (though you won't show them in the diagram).

3. Determine Dependencies:
   - Analyze the import sections and class fields to determine the dependencies between components.
   - This helps in understanding which components rely on others.

4. Organize Components:
   - Arrange the components logically in the diagram.
   - Use arrows to represent dependencies between components (e.g., one component depends on another).

Prompt Engineering – Persona Pattern

- ChatGPT eine Rolle zuschreiben: (Point of View)

> "You are going to pretend to be a software architect whose job is to reverse engineer projects in order to create diagrams of the software architecture. Don’t pay too much attention to implementation details but try to model the most important components."

Prompt Engineering – Template Pattern

- Struktur des Inputs erklären:

"Can you generate a component diagram using the following information including class descriptions (imports, extends, implements, package, methods, parameters) based on a Spring Boot project. The source code information is structured like this:

Class:<Classname>
implements:<Classnames>
extends:<Classnames>
Package:<Packagepath>
imports:<Imports>
Modifiers:<public/private/protected>
Field:<Fieldname: Type>
Method:<Method(parameters): Returntype>"

3. ChatGPT API

- **Model Version**
  - GPT 4

- **Nachrichten Länge**
  - Begrenzt auf max. **8000** Tokens
  - Aufteilung der Nachricht auf **Splits**
  - Warten mit der Bearbeitung, bis Erhalt aller Splits
  - => Schlechte Qualität

- **Temperature**
  - **Niedrig:** Für deterministische, konsistente Ergebnisse (z.B. 0.2).
  - **Hoch:** Für kreativere, variablere Antworten (z.B. 0.8).
  - Kein Human-In-The-Loop => **Niedrige Temperature**

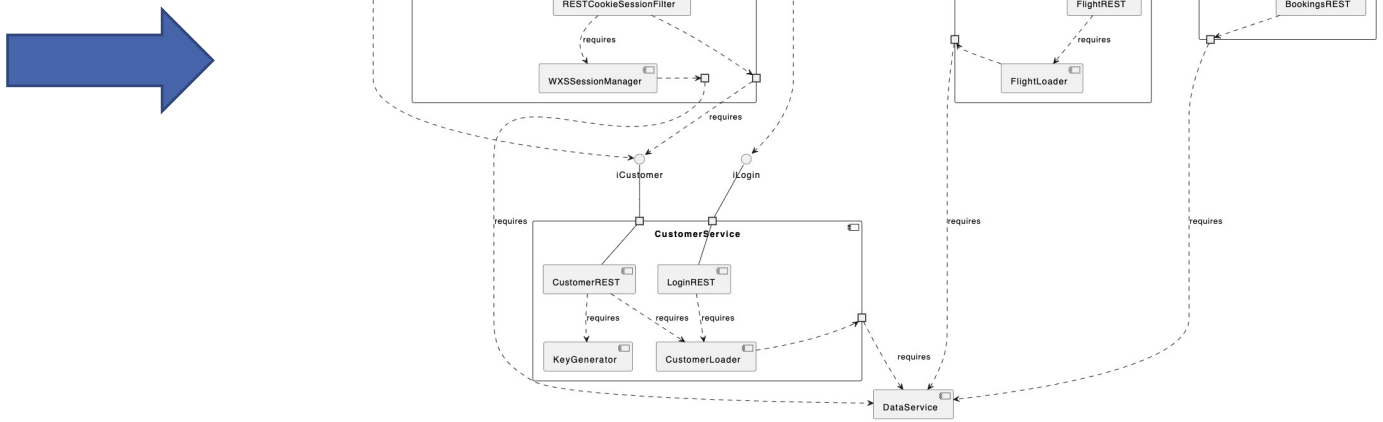
4. Visualisierung

- Plant UML: Textbasierte UML-Diagramme
- Einfache Rückgabeformat für LLM
- Viele Libraries verfügbar
- Öffentlich zugänglich

```
@startuml
class Example {
  +int id
  +String name
  +void method()
}

Example --> "0..*" OtherClass
@enduml
```

Ergebnisse

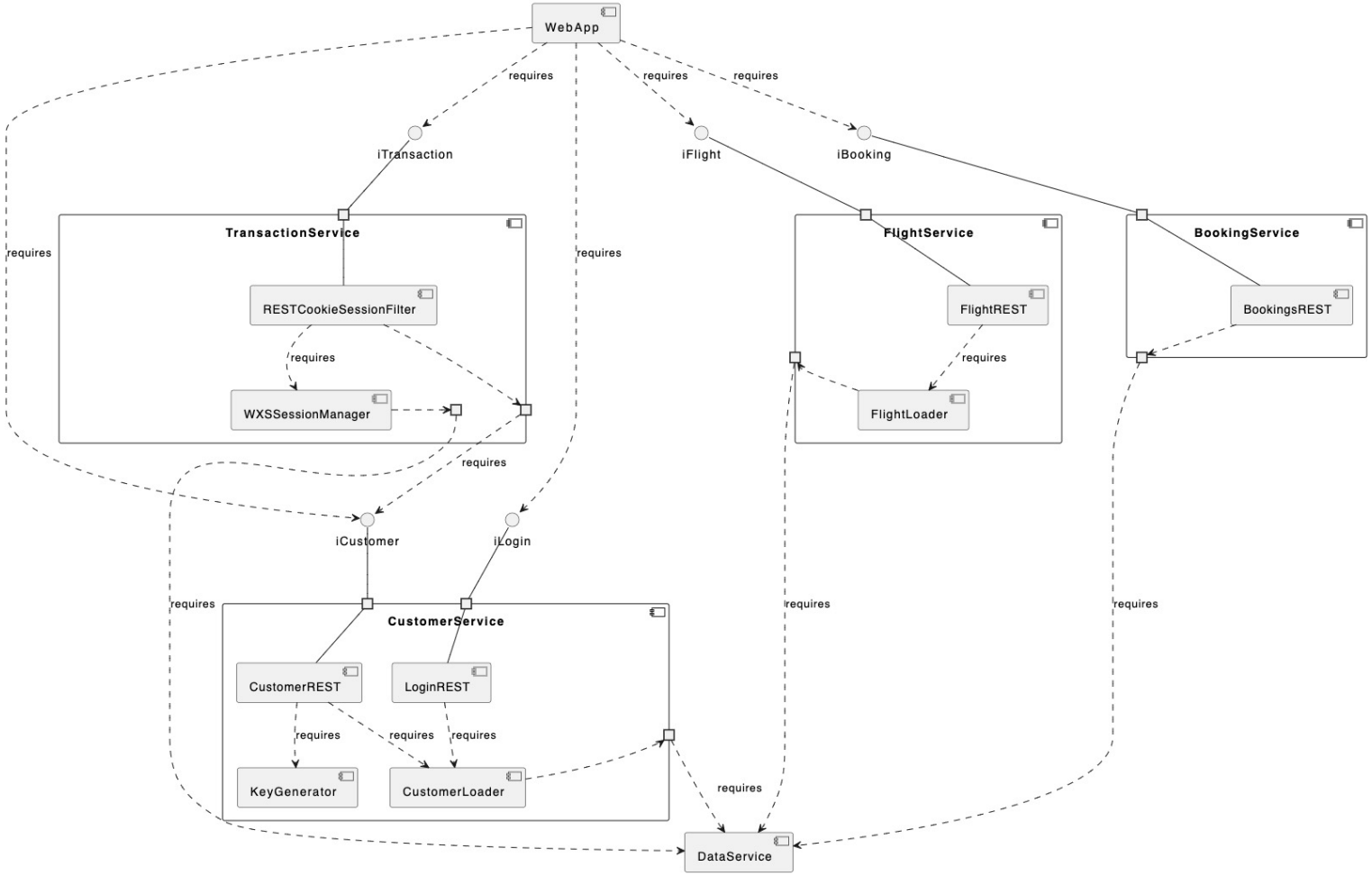
Feature Extraktion → Prompt Engineering → ChatGPT Interaktion → Visualisierung



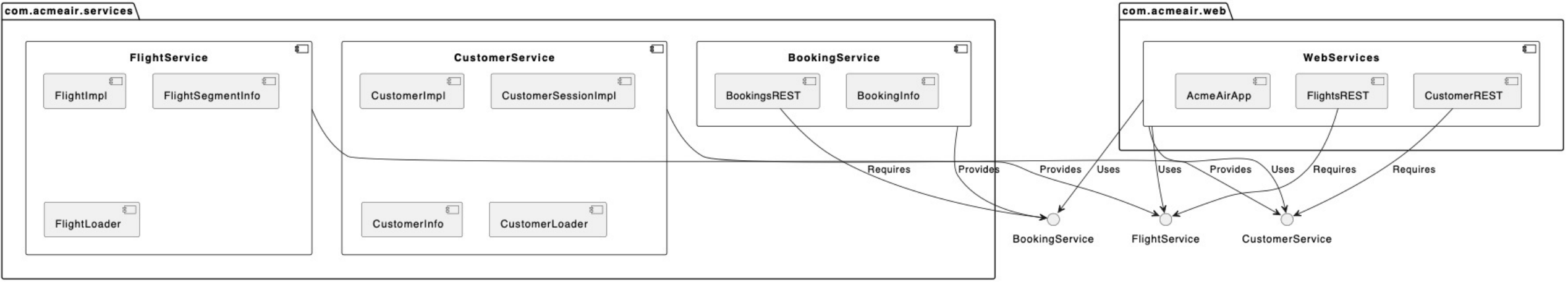
Next, I will create a component diagram based on the provided details. Let's generate the image with these details.

Here is the generated component diagram based on the information provided. If you need any adjustments or further details, feel free to let me know!

AcmeAir Gold Standard

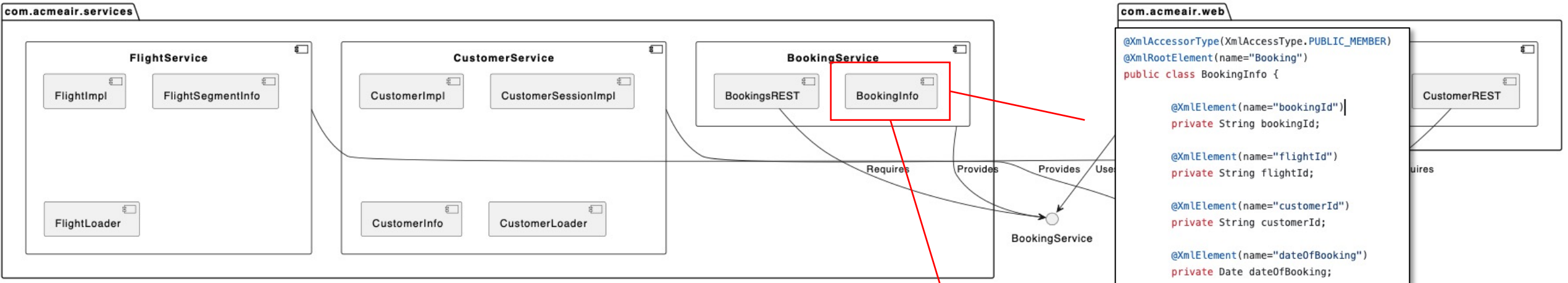


AcmeAir ChatGPT



- 4 Komponenten erkannt – Transaction Service fehlt

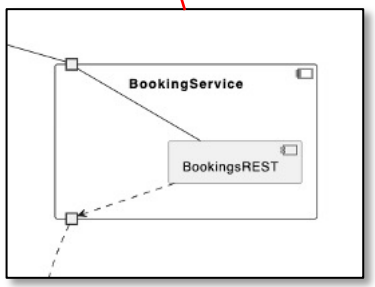
AcmeAir ChatGPT




```java
@XmlAccessorType(XmlAccessType.PUBLIC_MEMBER)
@XmlRootElement(name="Booking")
public class BookingInfo {

    @XmlElement(name="bookingId")
    private String bookingId;

    @XmlElement(name="flightId")
    private String flightId;

    @XmlElement(name="customerId")
    private String customerId;

    @XmlElement(name="dateOfBooking")
    private Date dateOfBooking;

    @XmlElement(name="pkey")
    private BookingPKInfo pkey;

    public BookingInfo() {

    }
```

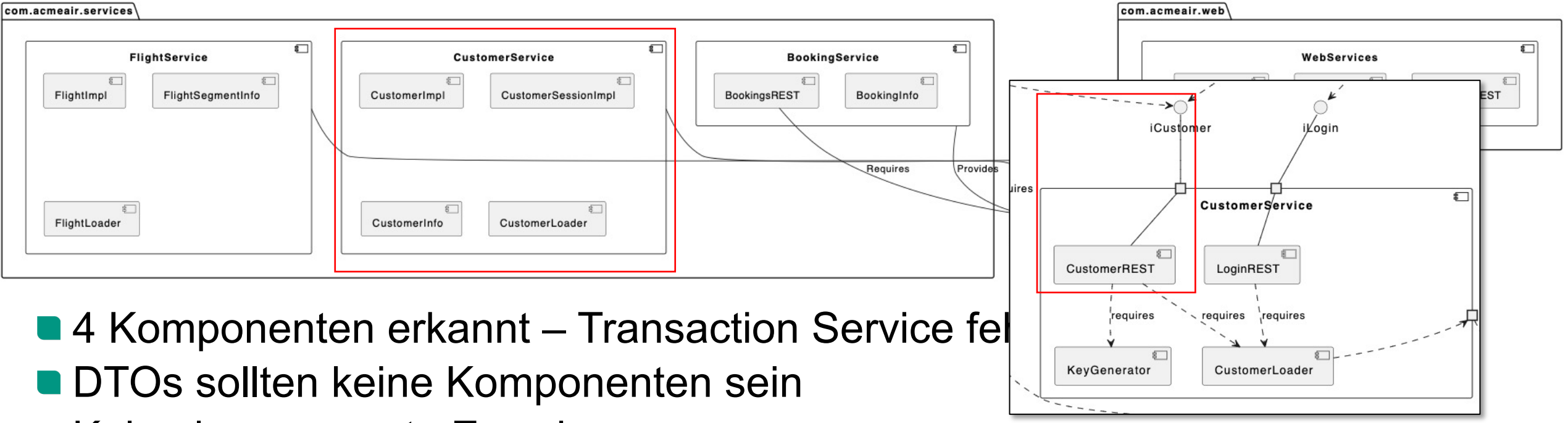
BookingInfo Code

- 4 Komponenten erkannt – Transaction Service fehlt
- DTOs sollten keine Komponenten sein



Gold Standard

AcmeAir ChatGPT



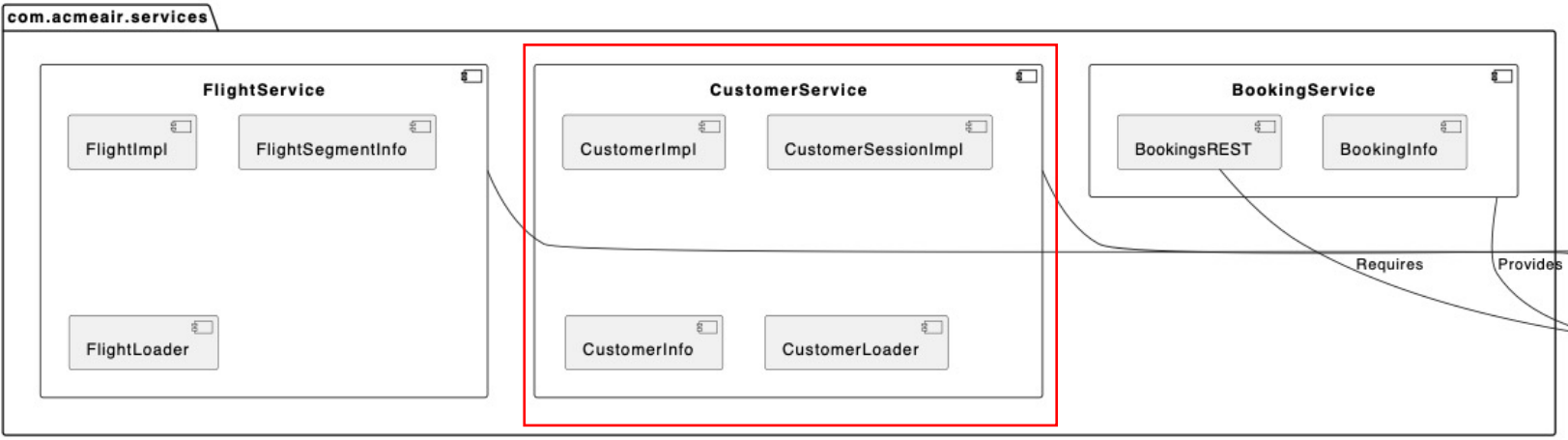
Gold Standard

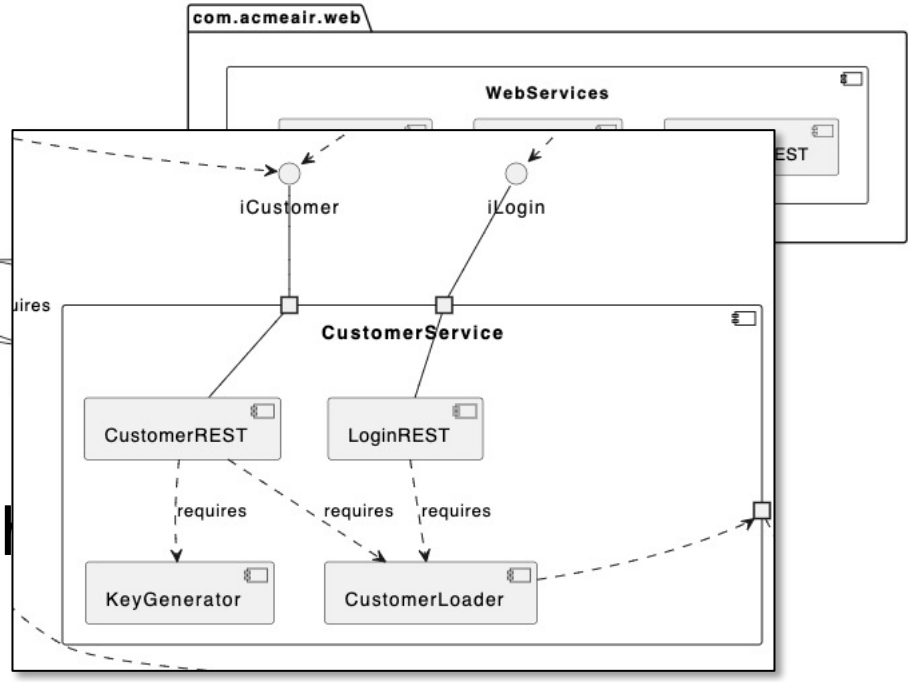
- 4 Komponenten erkannt – Transaction Service fe
- DTOs sollten keine Komponenten sein
- Keine konsequente Zuweisung

```
@Path("/customer")
public class CustomerREST {
```

CustomerRest Code

AcmeAir ChatGPT

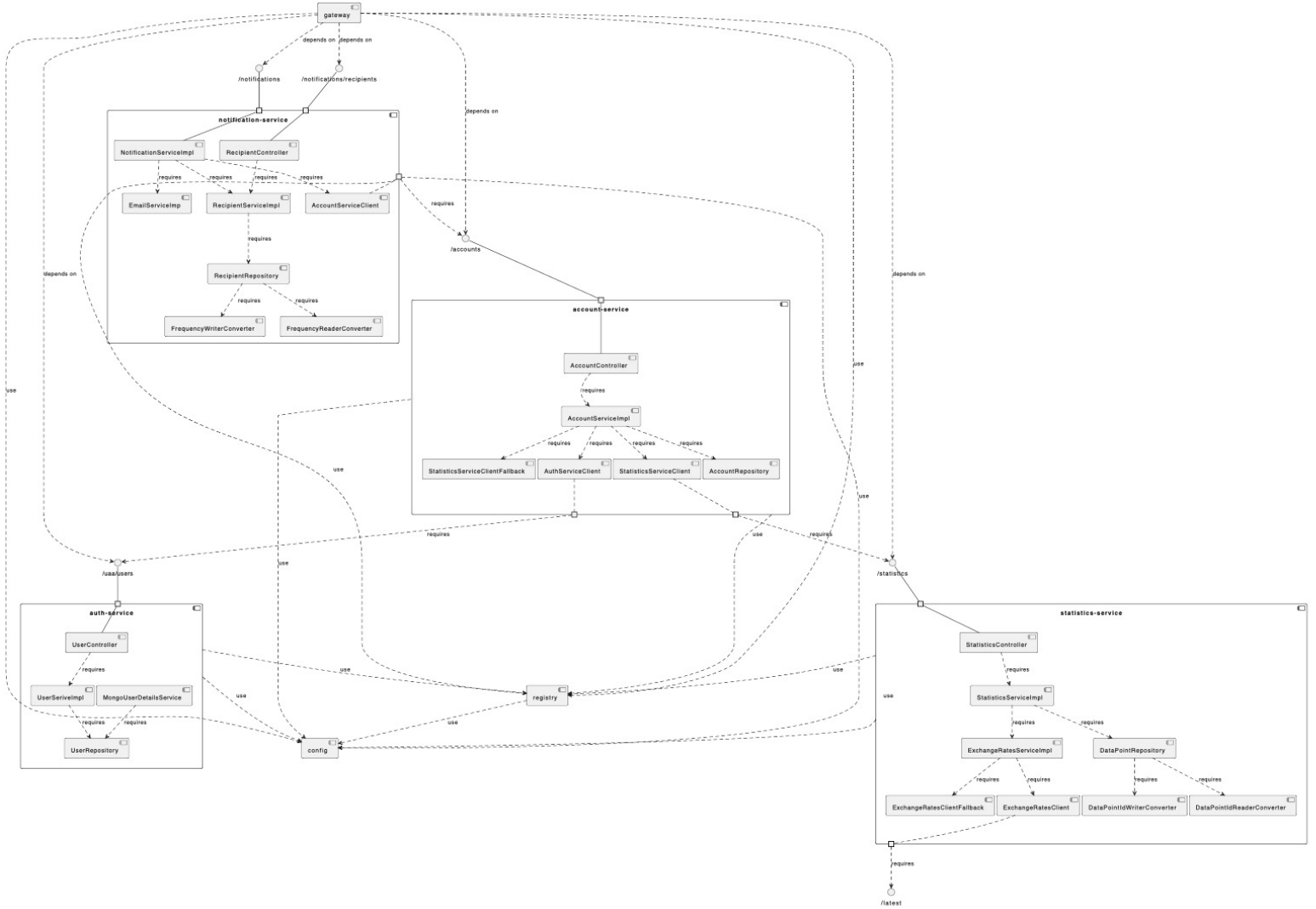




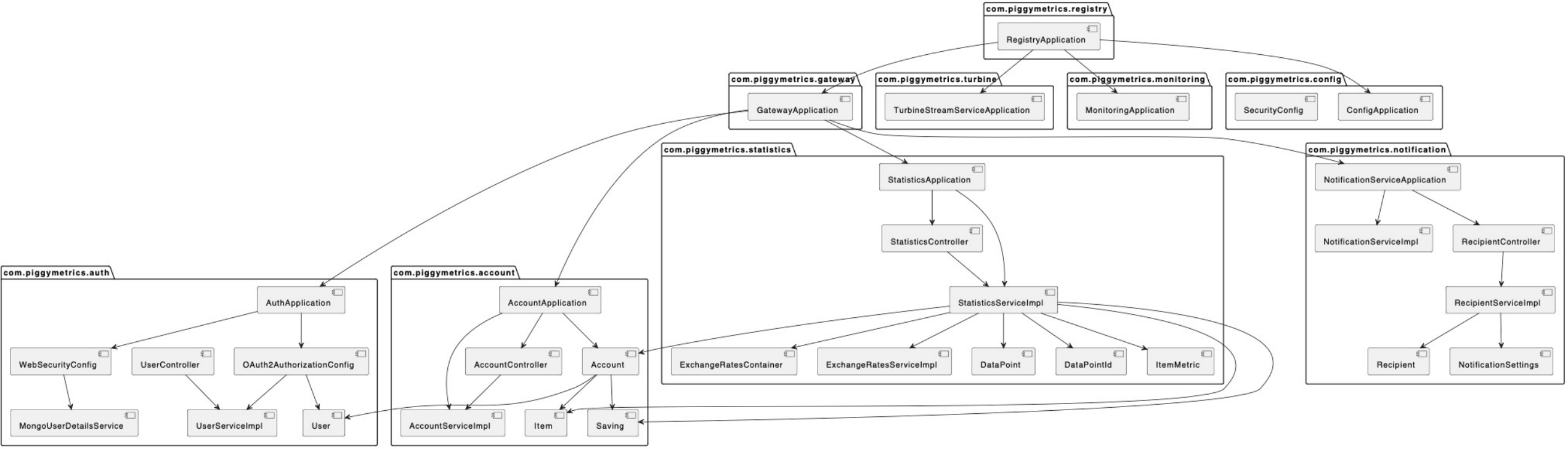
Gold Standard

- 4 Komponenten erkannt – Transaction Service fe
- DTOs sollten keine Komponenten sein
- Keine konsequente Zuweisung
- Fehlende Relationen zwischen Komponenten

PiggyMetrics Gold Standard

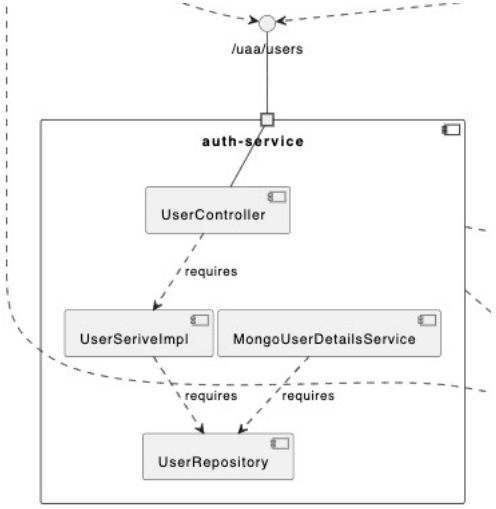


PiggyMetrics ChatGPT

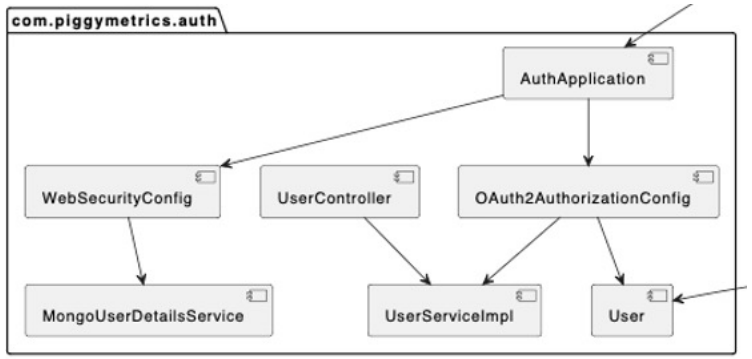


- Alle 4 wichtigen Komponenten erkannt

PiggyMetrics ChatGPT



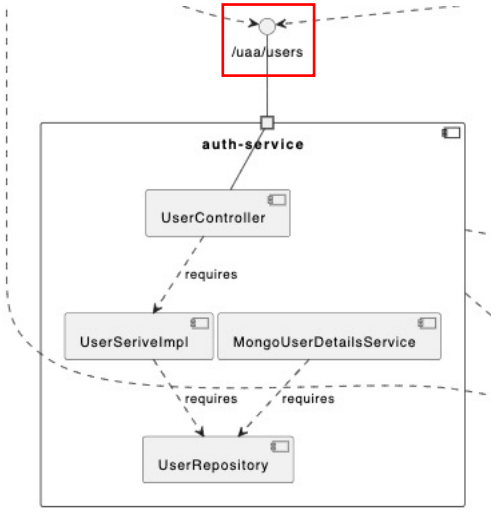
Gold Standard



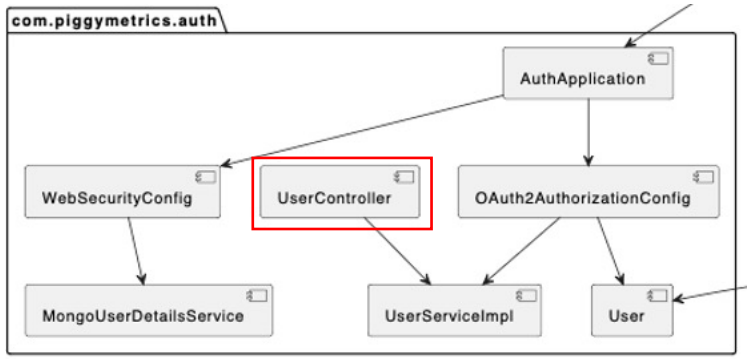
ChatGPT

- Keine unverbundenen Komponenten

PiggyMetrics ChatGPT



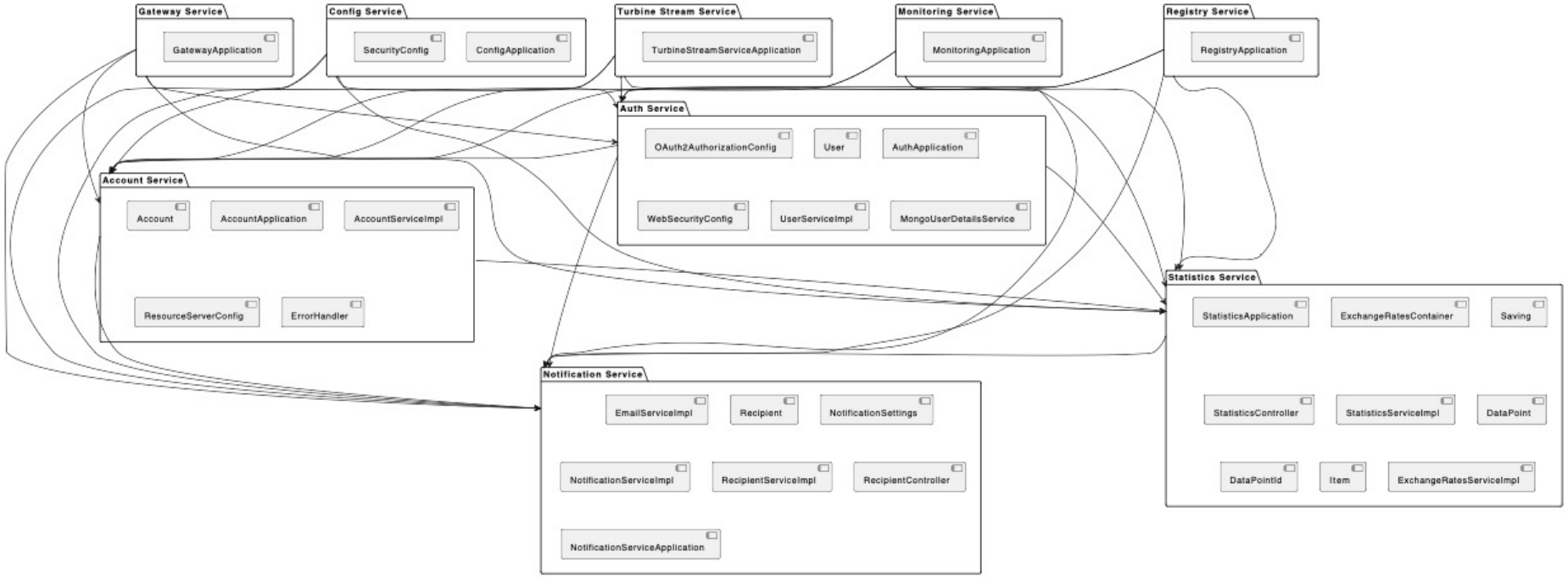
Gold Standard



ChatGPT

- Keine unverbundenen Komponenten
- Schnittstelle nicht erkannt

PiggyMetrics ChatGPT



- Neues Modell bei jeder Ausführung

Lessons Learned

- Richtige Temperature auswählen
- Aufteilung der Eingabe führt zu schlechteren Ergebnissen
- Initiale Ergebnisse nicht perfekt

Ausblick

- Unterstützung von traditionellen Reverse Engineering Ansätzen mit ChatGPT
- Neuere Versionen von ChatGPT

Vielen Dank für die Aufmerksamkeit!

Fragen?