Technische Universität München

COMMERCETOOLS

sebis

Konzeption und prototypische Realisierung einer einfachen, hochgradig skalierbaren

Multi-Mandanten-Architektur auf Basis der Google Cloud Services

Abschlussvortrag Bachelorarbeit

Sebastian Wenninger

17.10.2011

Betreuer: Prof. Dr. Florian Matthes

Agenda

1. Motivation
2. Kontext
3. Realisierung
4. Demo
5. Ausblick

COMMERCETOOLS

sebis

Über commercetools:

Die commercetools GmbH wurde 2006 gegründet und bietet ihren Kunden eine Software-as-a- Service eCommerce-Lösung an.

Kunden wie Red Bull, Burton und Brita betreiben internationale Shop-Lösungen auf der geclusterten Technologie.

Neben dem Hauptprodukt erweitert commercetools stetig das Produktsortiment und evaluiert verschiedene Plattform- und Cloud-Konzepte. commercetools hat seinen Sitz mit aktuell 25 Mitarbeitern in München.

1. Motivation

COMMERCETOOLS

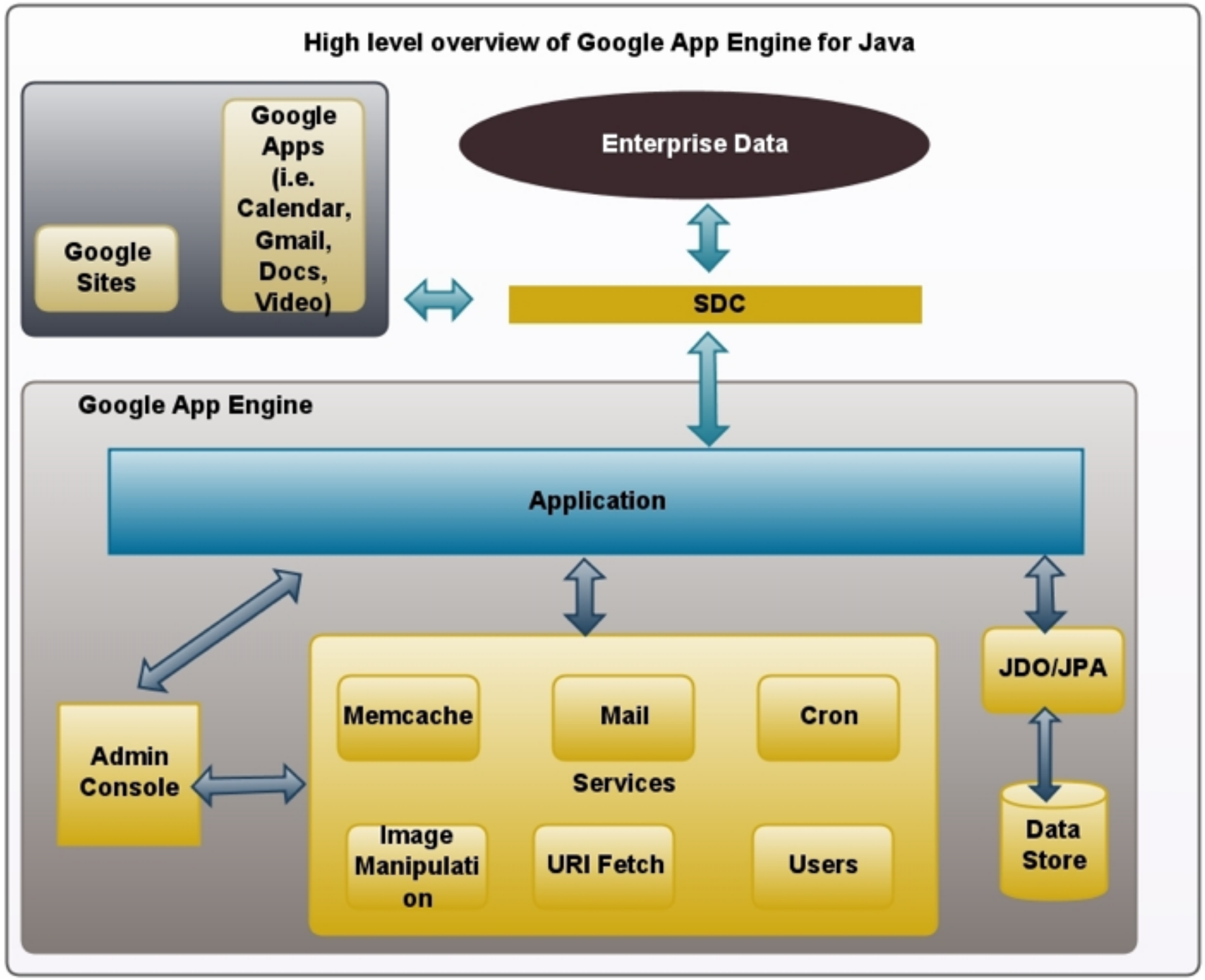
sebis

- iPhone-Shop Lösung vorhanden, aber kein Backend zur Verwaltung der Daten
- Ziel der Arbeit: Prototypische Entwicklung eines Backends
  - Anforderungen:
    - hoch skalierend
    - multi-tenant
    - nicht auf der eigenen Infrastruktur

➡ Nutzung von Platform-as-a-Service in der Cloud

➡ Evaluierung der Google-App-Engine (GAE) und der beteiligten Technologien

2. Kontext - GAE



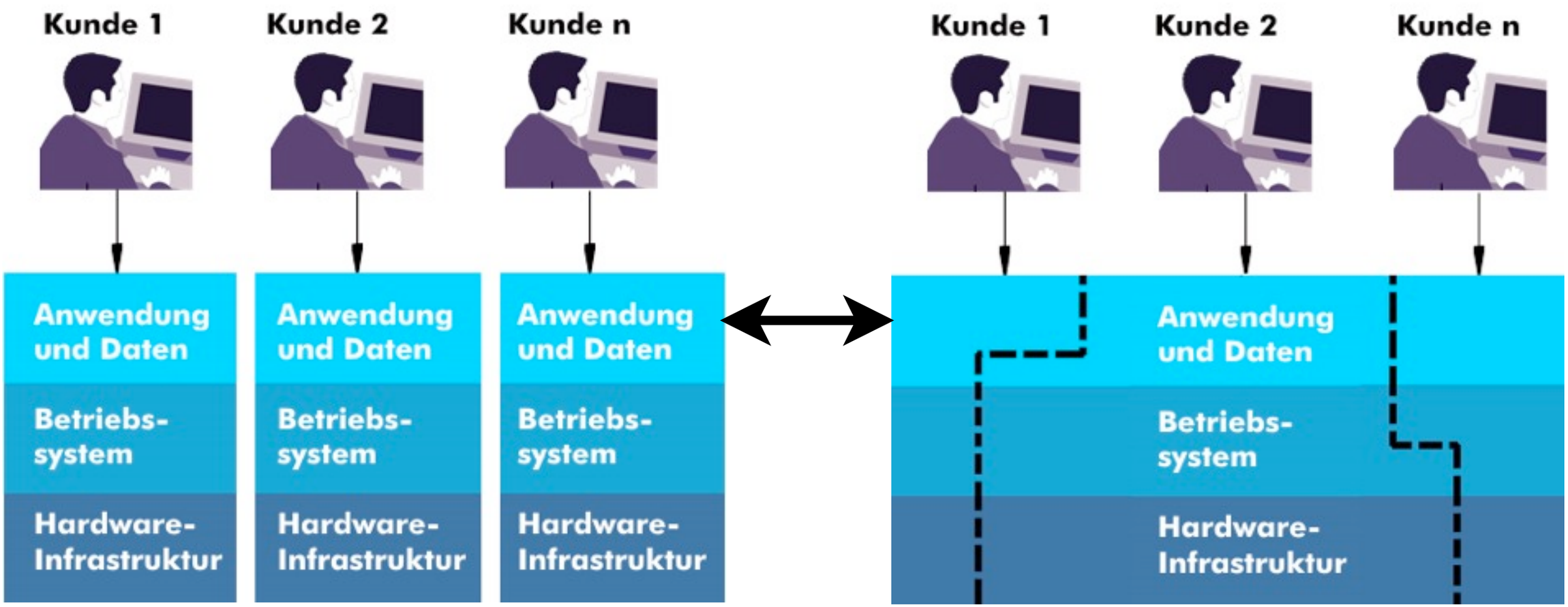
Quelle: http://www.byteonic.com/2009/overview-of-java-support-in-google-app-engine/

2. Kontext - Multi-Tenancy

sebis

Single-Tenant

Multi-Tenant

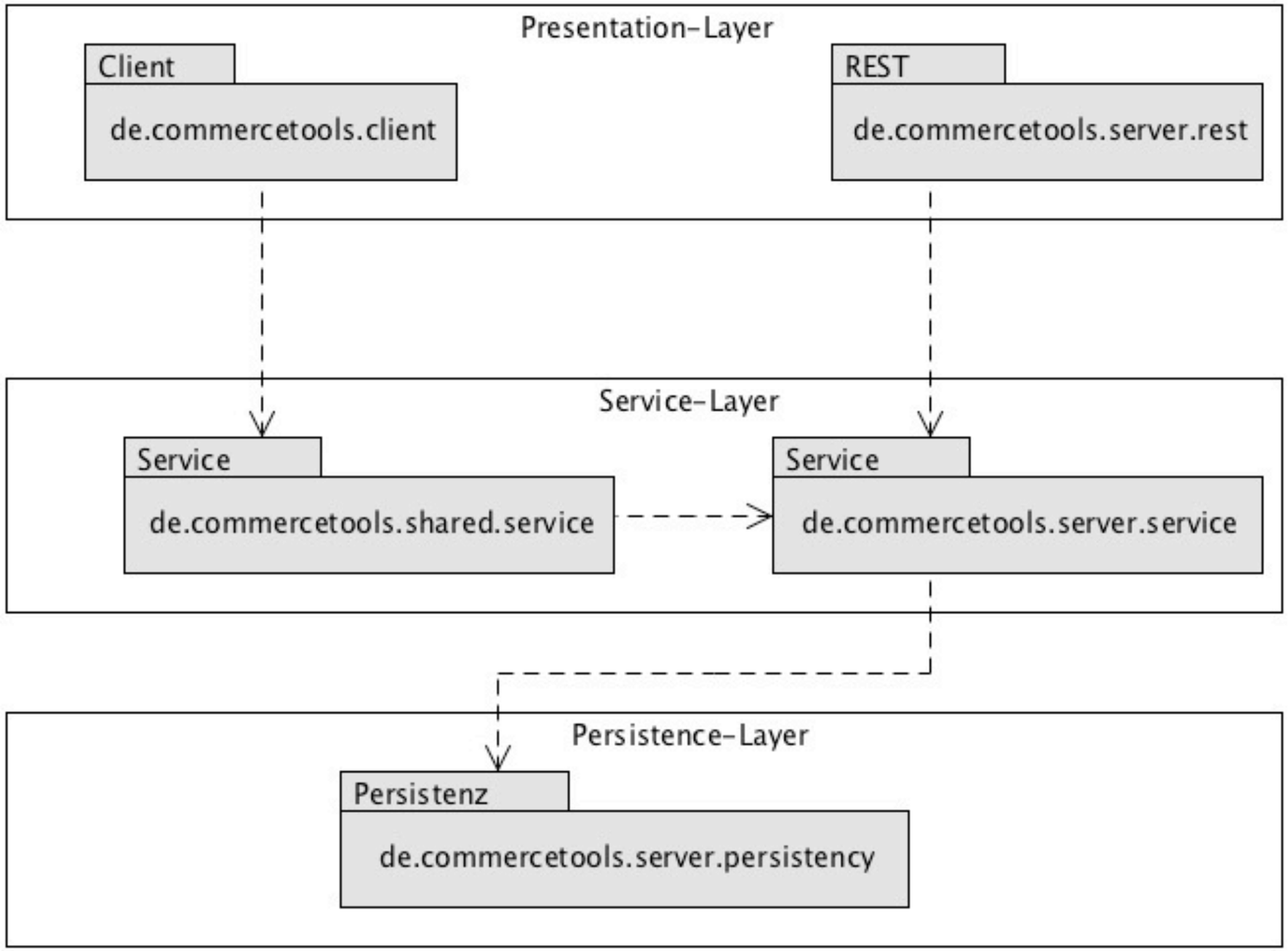


Quelle: http://www.itwissen.info/definition/lexikon/Multi-Tenancy-Architektur-multitenancy-architecture.html

3. Realisierung

sebis

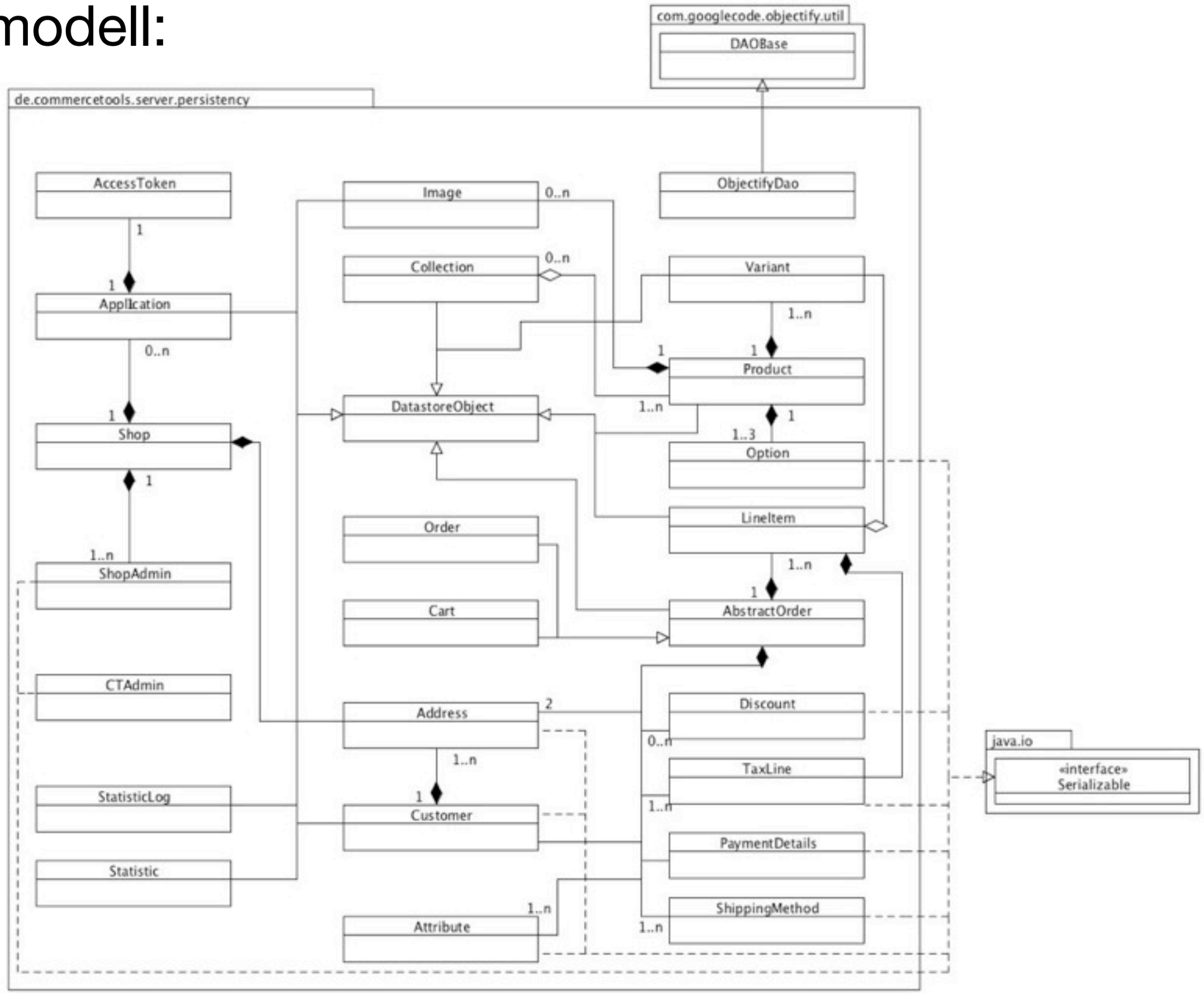
3-Schichten-Architektur:



3.1 Persistenzschicht

sebis

Datenmodell:



3.1 Persistenzschicht

sebis

Probleme:

- Einschränkungen des GAE-Datastore:
    - Entities < 1MB (z.B. bei Produktbildern)
    - Keine LIKE-Queries
- Umsetzung der Relationships, z.B. Produkte - Varianten

3.2 Applikationsschicht

sebis

- Interaktion mit der Persistenzschicht in Entity-spezifischen Services
- CRUD + Validierung

Probleme:

- Kommunikation mit User Interface (GWT-RequestFactory) über Proxy-Interfaces
  - ➡ Bei Änderungen manuelle Synchronisation von Code notwendig
- Automatische Validierung durch die RequestFactory
- Referenzielle Integrität

3.3 Präsentationsschicht

sebis

Zwei Ausprägungen:

- **Webservice (Restlet):**

  - Kommunikation über JSON-Objekte

  - Absicherung durch OAuth mit Bearer-Tokens + HTTPS

- **AdminPanel, UserPanel (GWT)**

  - Programmierung in Java, keine größeren HTML / Javascript / CSS Kenntnisse notwendig

  - Model-View-Presenter (Activities & Places)

3.3 Präsentationsschicht

sebis

Probleme:

- Komplizierte Dokumentation von Restlet (nur Minimalbeispiele)
- Eigene OAuth-Implementierung (+Wechsel der Tokens)
- GWT Aufwändig zu testen
- Kommunikation mit der Applikationsschicht nicht trivial (RequestFactory)
- Einarbeiten in Activities & Places
- Aufteilung der Website in mehrere Teilbereiche mit Activities & Places

4. Demo

sebis

DEMO

5. Ausblick

sebis

Kurzfristig:

- Implementierung einer Warenkorb-Logik
- Steigerung der Performance

Langfristig:

- Entwicklung von Standard-Applikationen
- Kompletter Shop-Baukasten (evtl. inklusive CMS)

Q & A

sebis

Vielen Danke für Ihre Aufmerksamkeit.

Fragen?