РАЗПРЕДЕЛЕНИ ПРИЛОЖЕНИЯ

Павел Кюркчиев
Ас. към ПУ „Паисий Хилендарски“
@pkyurkchiev

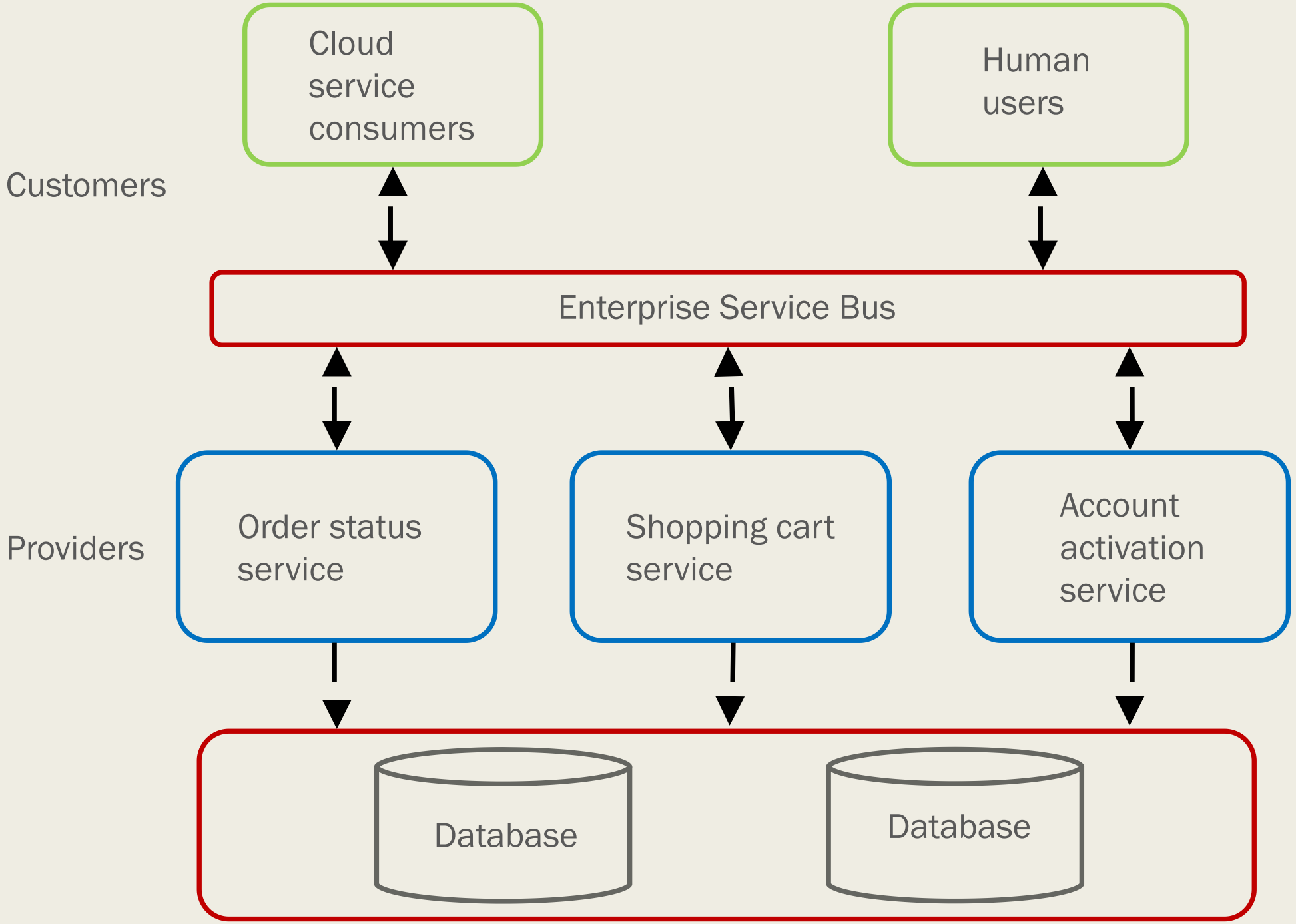
РАЗПРЕДЕЛЕНИ АРХИТЕКТУРИ SOA

Service-oriented architecture (SOA)

- SOA е стил на софтуерен дизайн, при който услугите се предоставят от софтуерни компоненти чрез комуникационни протоколи в интернет мрежата.

SOA основни компоненти

- Service provider - Той създава услугата и предоставя информацията за нея към съответния регистър (предоставяйки достъп до нея).
- Service broker, service registry or service repository - Основната му функция е да предостави информацията за уеб услугата на всеки потенциален клиент.
- Service requester/consumer - Това са всички клиенти, които достъпват и използват информацията, предоставена от услугите.


Cloud service consumers
Human users
Customers
Enterprise Service Bus
Providers
Order status service
Shopping cart service
Account activation service
Database
Database

SOA и Web 2.0

Web 2.0

- User generated content
- Services Model
  - *Сървис ориентира архитектура*
- Поява на RSS, Web API ...

SOA услуги

- Услугите са градивният блок на SOA архитектурата.
- Те могат да бъдат изградени от други услуги.
- За потребителя те представляват черна кутия (той не знае как работят).
- Услугите представят бизнес логиката на приложението.

Характеристики на услугите

- Автономни – независими са от останалата част на системата.
- Stateless – не знаят за предишни изпратени от вас заявки (не пазят състоянието на заявката).
- Приемат заявка и връщат отговор.
- Добре описани са.
- Използват стандартизирани интерфейси.

Характеристики на услугите

- Характеристики на услугите
- Комуникацията се осъществява през стандартни протоколи:
  - *HTTP, FTP, SMTP, RPC, MSMQ….*
  - *JSON, XML, SOAP, RSS, WS-….*
- Те са платформено независими.

Свойства на услугите

- Услугата представлява бизнес активност със специален изход (резултат).
- Услугата е самостоятелна, самосъдържаща се.
- Услугата е черна кутия за ползвателите.
- Услугата може да се състои от други, скрити отдолу, услуги.

Принципи на SOA – препокриват тези на услугите

- Стандартизирани условия на услугите.
  - *Услугите се придържат към стандартни комуникационни условия, които се дефинират от един или повече документи, описващи услугите за дадения набор от уеб услуги.*
- Автономност на връзката между услугите (част от loose coupling).
  - *Връзката между услугите е минимализирана до ниво, в което те знаят само за тяхното съществуване.*

- Прозрачност на местоположението на услугите (част от loose coupling).
  - *Услугите могат да бъдат извикани от всякъде в мрежата, без значение къде се намират.*
- Дълголетие на услугата
  - *Услугите могат да се проектират така, че да имат дълъг живот. Където е възможно, услугите трябва да избягват наложени от ползвателите форми на промяна на бизнес логиката им.*
  - *В сила трябва да е следното правило: "Ако извикаш една услуга днес, трябва да можеш да извикаш същата услуга и утре".*
- Абстрактни услуги
  - *Услугите трябва да работят като черна кутия, като по този начин тяхната вътрешна логика е скрита от ползвателите.*

- Автономни услуги
  - *Услугите са независими и контролират функционалностите, които се скриват, както по време на Design-time (времето за разработка), така и по време на run-time (времето за действие).*
- Услуги без състояние
  - *Услугите нямат състояние, те или връщат положителен резултат, или връщат изключение (exception). Едно извикване на услугата не трябва да влияе на друго такова.*
- Детайлност на услугата
  - *Принцип, който цели да подсигури адекватния размер и обхват на услугата. Функционалностите, предоставени на ползвателя от услугата, трябва да бъдат уместни.*

- Нормализиране на услуга
  - *Услугите се декомпозират и консолидират (нормализират), за да се намали излишното. За някои услуги това не може да се направи. Това са случаите, в които бързодействието, лесният достъп и агрегацията са нужни.*
- Композитност на услуги
  - *Услугите могат да се използват, за да се композират други услуги.*
- Откриване на услуги
  - *Услугите имат специални комуникационни метаданни, чрез които услугите лесно могат да се откриват и да имат достъп.*

- Преизползване на услуги
  - *Логиката е разделена на малки парчета, което позволява създаването на отделни малки услуги и тяхното лесно преизползване.*
- Енкапсулация на услуги
  - *Много услуги, които в началото не са планирани като SOA, могат да се енкапсулират (обвият) и да станат част от SOA архитектура.*

Подходи на имплементация и технологии

- Web services based on WSDL and SOAP
- Messaging, e.g., with ActiveMQ, JMS, RabbitMQ
- RESTful HTTP, with Representational State Transfer (REST) представляващ собствен архитектурен стил, базиран на ограничения.
- OPC-UA
- WCF (Microsoft's implementation of Web services, като част от WCF)
- Apache Thrift
- SORCER

ВЪПРОСИ ?