UA/DETI • 41951: Análise de Sistemas

David Pelicano (113391), Pedro Melo (114208)(pivot), Rúben Pequeno (102480), Simão Almeida (113085)

Grupo 501, v2024-03-20.

RELATÓRIO

5 Lab: Vistas de arquitetura

Exercício 5.1

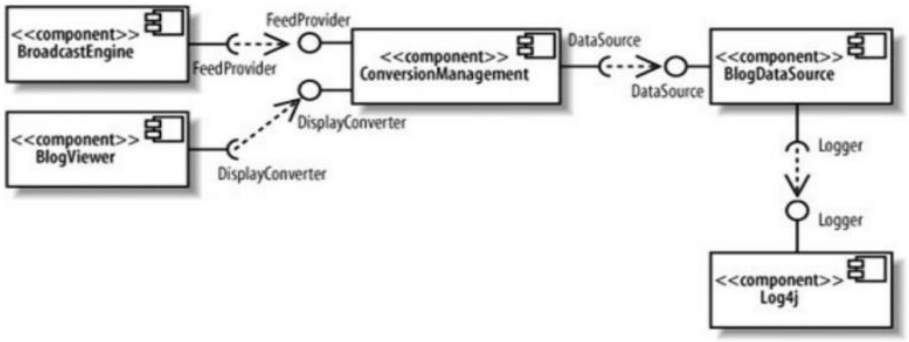


Diagrama 1: Componentes de uma aplicação (hipotética) de gestão de blogs[1].

a)
O Diagrama 1 representa os componentes de uma aplicação hipotética de gestão de blogs. A aplicação é composta por vários componentes interconectados que trabalham em conjunto para fornecer, converter e apresentar dados.

O **BroadcastEngine** alimenta dados para o **FeedProvider**, que por sua vez está ligado ao **ConversionManagement**. Este último componente está associado a um **DisplayConverter**. Os dados são então passados para o **BlogViewer** para visualização.

Paralelamente, o **ConversionManagement** também está conectado a um **DataSource** ligado ao **BlogDataSource**, que por sua vez se liga a um sistema de registo **Logger** usando o componente **Log4j** para manter registos detalhados das operações.

b)
Apache Log4j é uma estrutura versátil de registo Java de nível industrial composta por uma API, a sua implementação e componentes auxiliam a implantação para vários casos de uso. Log4j é usado por 8% do ecossistema Maven e listado como um dos 100 projetos de software de código aberto mais críticos.

Algumas das características destacadas do Log4j:

- Inclusão de baterias: Log4j inclui um rico conjunto de componentes para auxiliar vários casos de uso. Appenders direcionados a arquivos, soquetes de rede, bancos de dados, servidores SMTP, etc. Layouts que podem renderizar CSV, HTML, JSON, Syslog, etc. saídas formatadas. Filtros que

podem ser configurados usando taxas de eventos de log, expressões regulares, scripts, tempo, etc. Lookups para acessar propriedades do sistema, variáveis de ambiente, campos de eventos de log, etc.

- Separação da API: A API para Log4j (ou seja, log4j-api) é separada da implementação (ou seja, log4j-core) tornando claro para os desenvolvedores de aplicativos quais classes e métodos eles podem usar enquanto garantem compatibilidade futura.
-
- Sem bloqueio de fornecedor: Embora a API Log4j seja implementada pelo Log4j em sua totalidade, os usuários podem optar por usar outro backend de registo.

Para incluir o Apache Log4j em um projeto usando a ferramenta de construção Gradle, adiciona-se as seguintes dependências ao seu arquivo build.gradle:

```
dependencies {

    implementation 'org.apache.logging.log4j:log4j-api:2.23.1'

    implementation 'org.apache.logging.log4j:log4j-core:2.23.1'

}
```

Estas coordenadas identificam a versão 2.23.1 do Log4j no repositório Maven. A primeira linha adiciona a API Log4j ao projeto e a segunda linha adiciona a implementação do núcleo Log4j.

Exercício 5.2

a) Um dos exemplos concretos de uma arquiterura multicamada encontrada na internet é a Shopify.

UI Layer (Camada da interface do usuário)

Shopify oferece uma interface de administração e uma interface de loja online . A interface de administração permite que os comerciantes gerenciem os produtos,pedidos,clientes, entre outras funcionalidades destes. A interface da loja online é o ponto de interação com os clientes, onde estes podem navegar pelos produtos,fazer compras e interagir com a loja. Para além disto, Shopify oferece uma interface de website como é habitual existir, onde inclui o layout e design que são públicos para o cliente.

Domain Layer (Camada do domínio)

Nesta camada, estão as principais funcionalidades relacionadas ao comércio eletrónico, como processamento de pedidos, cálculo de impostos e integrações de pagamento. Todas essas funcionalidades são parte do núcleo do negócio do Shopify e são encapsuladas nesta camada.

Camada da base de dados

Esta empresa lida com grandes volumes de dados, incluíndo informações de produtos, detalhes de pedidos, informações dos clientes e muito mais. Estes dados são armazenados num banco de dados robusto e escalável utilizando tecnologias como bancos de dados e NoSQL (BackEnd).

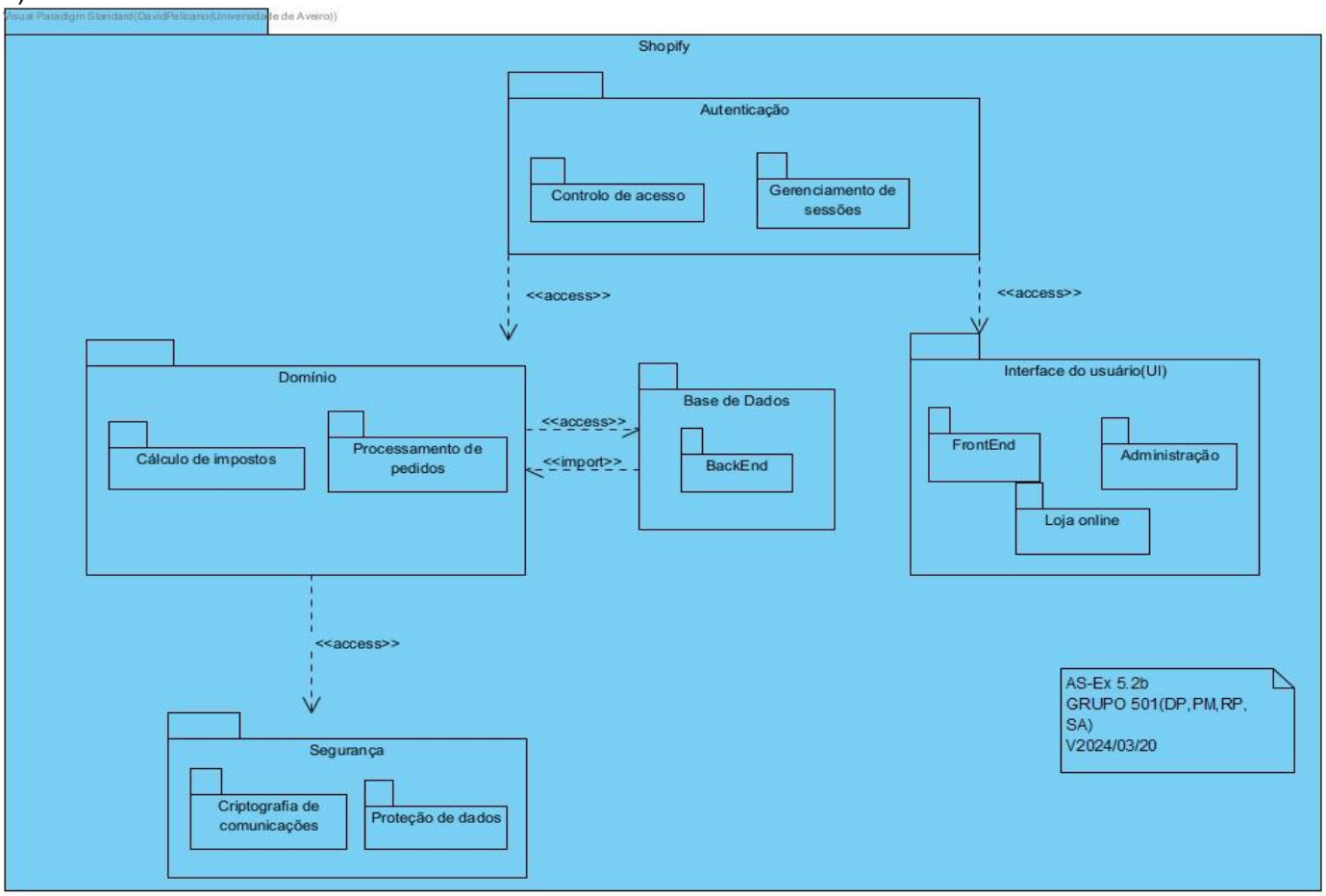
Camada de autenticação

Esta camada trata da autenticação e autorização dos usuários. Esta inclui funcionalidades como login, registo de novos usuários, gerenciamento de sessões e controlo de acesso às diferentes partes da aplicação.

Camada de Segurança

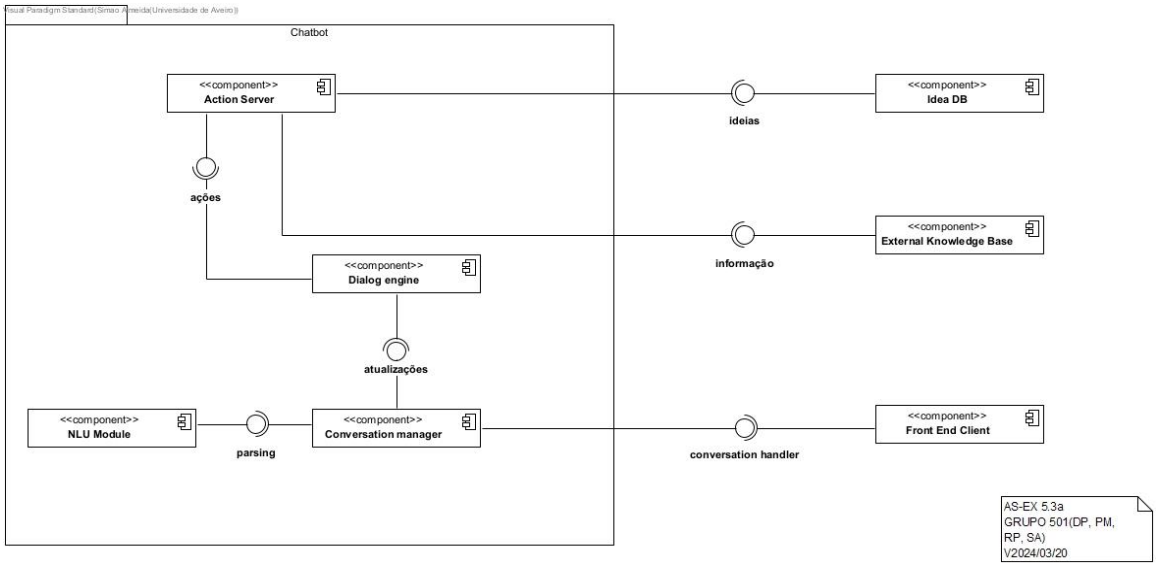
Esta camada trata das medidas de segurança da aplicação, incluindo proteção contra ameaças como ataques de “hackers”, proteção de dados sensíveis, criptografia de comunicações

b)



Exercício 5.3

A)



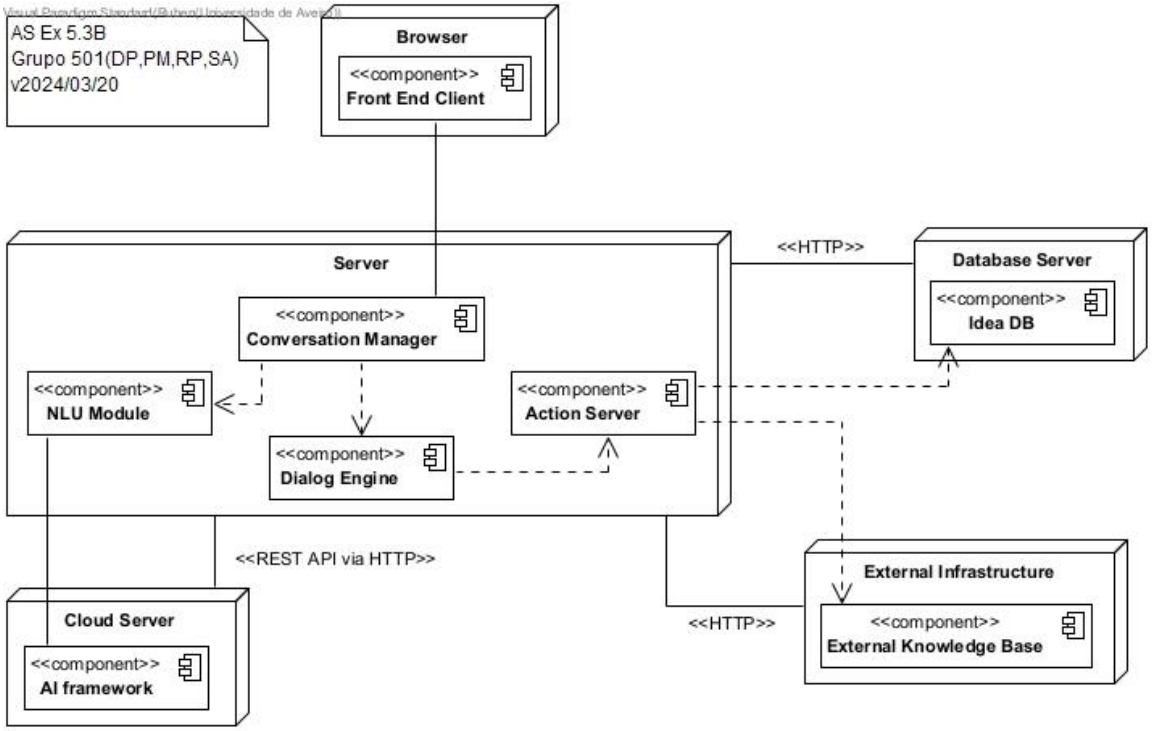
Neste diagrama é possivel ver 7 componentes. Os componentes **Idea DB**, **External Knowledge Base** e **Front End Client** são clientes externos, enquanto os componentes **Action Server**, **Dialog Engine**, **Conversation manager** e **NLU Module** pertencem ao módulo Chatbot.

O **Idea DB** fornece ao **Action Server** ideias, enquanto o **External Knowledge Base** fornece informação.

O **Action Server** realiza ações ao **Dialog Engine**.

O **NLU Module** faz o "parsing" para o **Conversation Manager**. Este fornece atualizações ao **Dialog Engine** e faz o "conversation handler" para o **Front End Client**.

B)



O diagrama de instalação apresentado acima representa a conectividade entre os componentes do sistema de "chat bot".
Os utilizadores finais acedem a partir de um browser através do componente ***Front End Client*** com ligação ao ***Conversation Manager***.

O componente ***Action Server*** presente no nó Server, acede tanto à componente ***Idea DB*** no servidor da Database, como à base de conhecimento externa através de ligações HTTP.

O ***módulo NLU*** pode comunicar com a Cloud através do REST API via HTTP para aceder aos serviços de apoio disponiveis em *frameworks* de Inteligência Artificial.