Ciências ULisboa | Informática

Projeto de Tecnologias de Redes

Enunciado do Projeto

Conceção, desenho e concretização de um sistema distribuído

---

1 Introdução

Este projeto tem como objetivo colocar em ação as boas práticas na conceção, desenho e concretização de um sistema distribuído, tal como foram estudadas em unidades curriculares anteriores da área de Redes e Sistemas. Este objetivo abarca as seguintes áreas:

- Conceção, definição de casos de uso, definição de uma API REST, definição de requisitos não-funcionais
- Desenho
- Concretização

São também avaliadas as metodologias, ferramentas, bibliotecas e serviços externos usadas no projeto.

2 Casos de uso e API REST

O sistema distribuído a desenvolver deve oferecer uma interface para aplicações (API) do tipo REST com acesso a todas as funcionalidades em dependência de um procedimento robusto de autenticação e de autorização.

3 Requisitos não-funcionais

A especificação do sistema deve incluir a forma como vão ser satisfeitos os requisitos não-funcionais ao nível da arquitetura ou ao nível da implementação. O projeto deve atender aos seguintes requisitos não-funcionais:

Requisito	**Especificação**
Fonte aberta (*open source*)	O sistema deve ser de código fonte aberto.
Ambiente de desenvolvimento	O ambiente de desenvolvimento deve fazer uso de ferramentas de controlo de versões que suportem uma equipa de programadores. Adicionalmente devem ser usados mecanismos de automação que facilitem o desenvolvimento da aplicação.
Lançamento para produção (*deployment*)	O sistema deve ter um automatismo que permite lançá-lo em produção a partir de um repositório de código.
Interoperabilidade	O sistema deve ter uma API (REST, por exemplo) que lhe permite interagir com outras aplicações.
Escalabilidade	O sistema deve ter uma capacidade ajustável para atendimento de pedidos. Pode ser horizontal através da inclusão de mais nós e/ou vertical através de adicionar mais recursos aos nós existentes. A escalabilidade horizontal deve contar com balanceamento de carga.
Elasticidade	É desejável que o sistema possua capacidade de elasticidade. Embora este não seja um requisito obrigatório para o projeto, será valorizado na avaliação.
Tolerância a faltas	O sistema deve tolerar a falha de qualquer dos seus componentes, através de principalmente oferecer redundância ativa.

Disponibilidade	O sistema deve responder a qualquer pedido dentro de um tempo razoável pré-definido.
Tempo de resposta e desempenho	O sistema deve ter um tempo de resposta e um desempenho adequados à expetativa dos utilizadores.
Nome de domínio	Deve ser criado e registado um nome de domínio para obtenção do certificado.
Segurança - Canais seguros	Obtenção de um certificado.
Segurança – Autenticação e autorização	O sistema computacional e as redes devem estar cobertas por políticas de segurança que fazem uso dos mecanismos de proteção adequados de modo a impedir o acesso não-autorizado a recursos e dados e também impedir a disrupção do serviço.
Privacidade	O sistema deve estar de acordo com os regulamentos para a privacidade dos dados (RGPD).
Segurança - Configuração de firewall	O sistema computacional e as redes devem estar cobertas por políticas de segurança que fazem uso dos mecanismos de proteção adequados de modo a impedir o acesso não-autorizado a recursos e dados e também impedir a disrupção do serviço.
Integridade dos dados	A infraestrutura deve assegurar a consistência dos dados.
Salvaguarda de dados	Os dados devem ser salvaguardados regularmente através de uma política adequada à importância dos dados e configurações. Adicionalmente deve estar definido um procedimento de restauro de qualquer parte dos dados.
Custo	Ponderando o custo das várias soluções, deve ser escolhida uma configuração que preste o melhor serviço pelo preço mais baixo. A escolha deve ser justificada tendo por base as várias opções.
Testes - carga e vulnerabilidade	O sistema deve ser desenvolvido com uma política integrada de testes que permitem ir testando à medida que se adicionam novos componentes ou que se alteram componentes existentes.

A implementação destes requisitos não funcionais será avaliada ao longo do curso em reuniões de avaliação periódica, conforme abaixo especificado:

1ª Reunião de Avaliação: **1ª semana de março/2024***

Design	1	Especificação API usando o padrão OpenAPI, em formato YAML ou JSON
	2	Especificar requisitos NF - escalabilidade
	3	Especificar requisitos NF - segurança
	4	Especificar requisitos NF - tolerância a faltas
	5	Design da arquitetura distribuída
	5.1	Aplicação do conceito em relação aos *load balancers*
	5.2	Aplicação do conceito em relação aos *web servers*
	5.3	Aplicação do conceito em relação às bases de dados
	6	GitHub - uso com *branches* individuais e *releases*

2ª Reunião de Avaliação: **1ª semana de abril/2024***

Arquitetura distribuída e tolerância a faltas	7	Implementação da arquitetura distribuída (implementação de instâncias e configuração de redes)
	7.1	Implementação dos balanceadores de carga
	7.2	Implementação dos servers

	7.3	Implementação das bases de dados
	8	Configuração dos balanceadores de carga e mecanismos de escalabilidade
	9	Configuração de mecanismos de tolerância a faltas e verificação de saúde
	9.1	Teste de tolerância a faltas nos balanceadores de carga
	9.2	Teste de tolerância a faltas nos webservers
	9.3	Teste de tolerância a faltas nas bases de dados

3ª Reunião de Avaliação: **1ª semana de maio/2024***

Segurança	10	Implementação de mecanismo de Autenticação e autorização (Auth0)
	11	Canais seguros, DNS e configuração de firewall
	11.1	Uso de TLS com certificado assinado por uma Autoridade Certificadora (AC)
	11.2	Nome de domínio registado e associado ao IP estático
	11.3	Configurações de firewall
	12	Configurar uso de APIs externas
	13	Gestão de credenciais e IPs na instalação

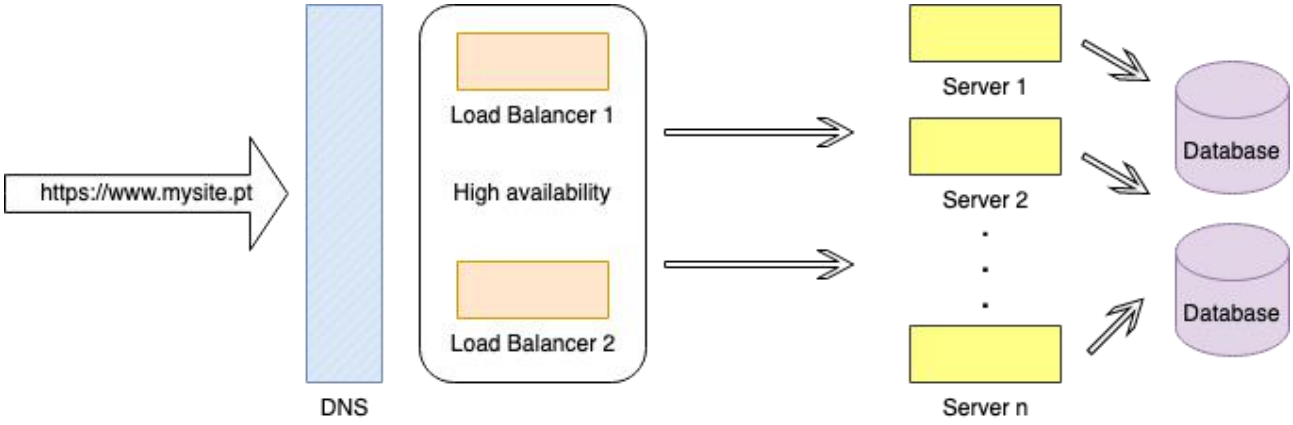
4ª Reunião de Avaliação: **1ª semana de junho/2024***

Automação e testes	14	Automação e scripts para construção e lançamento da aplicação
	15	Inclusão de testes unitários integrados ao processo de construção da aplicação (Github)
	16	Testes de aceitação com a API
	17	Testes de carga
	18	Testes de vulnerabilidades

* Os dias e a horários específicos das reuniões de avaliação serão agendados pelo professor e o grupo.

4 Desenho do sistema

Uma arquitetura sugerida é a seguinte:



É visível que a arquitetura deve incluir 3 tipos de componentes:

- Balanceador de carga
- Servidor

- Base de dados

Cada um destes componentes da arquitetura inclui múltiplas instâncias do balanceador de carga, do servidor aplicacional e da base de dados. **Cada uma destas instâncias deve estar instalada numa máquina virtual independente**.

O balanceador de carga deve estar configurado num modo de alta disponibilidade (*high availability*) para que não constituam um ponto único de falha. Existem duas configurações possíveis, ativo-passivo, onde um dos balanceadores realiza todo o trabalho e o 2º só entra em ação em caso de falha do 1º, e o ativo-ativo, onde ambos prestam serviço em simultâneo. **Neste projeto deverá ser utilizada a configuração ativo-passivo para alta disponibilidade nos balanceadores de carga.**

A arquitetura deve conter sub-redes distintas de acordo com a finalidade e a necessidade das ligações internas e externas, de modo a segmentar o tráfego e aumentar a segurança do sistema. Os balanceadores de carga, nomeadamente, devem estar conectados a duas sub-redes distintas: uma externa (por onde recebem ligações dos clientes) e uma interna (utilizada para os balanceadores se conectarem aos servidores).

Para o servidor, o número mínimo de instâncias é de duas. O número de réplicas deve poder ser escalado. Uma das formas de escalar é passiva, onde por execução de um procedimento são adicionadas novas réplicas do servidor e os seus endereços são adicionados à lista do balanceador de carga. Outra forma de escalar é ativa, neste caso o sistema monitoriza os níveis de carga e dinamicamente adiciona ou retira réplicas. Esta última forma confere uma propriedade de elasticidade ao sistema.

O sistema pode monitorizar o estado de cada uma das instâncias do servidor e automaticamente lançar novas instâncias sempre que alguma delas deixar de responder ou sempre que for necessário aumentar a capacidade do sistema para responder a um aumento do tráfego.

A base de dados é implementada com duas instâncias e o modo de redundância pode ser ativa-passiva.

É aconselhada a utilização de máquinas virtuais num dos fornecedores de serviços em nuvem. Neste caso, **os grupos deverão optar apenas pelas seguintes plataformas**: Google Cloud Platform (GCP); Amazon Web Services (AWS); e/ou Microsoft Azure. Desta forma o sistema pode ser acedido tanto pelos elementos do grupo como pelos docentes.

Não devem ser utilizados serviços de terceiros que ofereçam as capacidades de redundância, balanceamento de carga, escalabilidade e tolerância a faltas. Exemplos de serviços que não devem ser usados: Google AppEngine e Kubernetes. É esperado que seja cada grupo a concretizar estes requisitos não-funcionais usando apenas máquinas virtuais onde podem ser instaladas aplicações, eventualmente dentro de containers para facilitar o desenvolvimento e o lançamento em produção.

5 Implementação

Devem ser configuradas as *firewalls* de cada rede e máquina virtual.

6 Desenvolvimento do trabalho

O trabalho deverá ser desenvolvido gradualmente ao longo do semestre, atentando para os requisitos que serão avaliados em cada Reunião de Avaliação cujas datas estão especificadas na Seção 3.

7 Documentação a entregar

Cada grupo deverá entregar a documentação abaixo descrita ao longo da execução do projeto. Estas entregas correspondem à componente de avaliação periódica cujo valor total é de 10% da classificação.

	Pesos
1ª Reunião Periódica **Documentos a entregar:** - Especificação API usando o padrão OpenAPI, em formato YAML ou JSON. - Relatório referente aos itens 1 a 6 (e seus subitens) especificados para a 1ª Reunião Periódica na Seção 3 deste enunciado.	**2.5%**
2ª Reunião Periódica **Documentos a entregar:** - Relatório referente aos itens 7 a 9 (e seus subitens) especificados para a 2ª Reunião Periódica na Seção 3 deste enunciado.	**2.5%**
3ª Reunião Periódica **Documentos a entregar:** - Relatório referente aos itens 10 a 13 (e seus subitens) especificados para a 3ª Reunião Periódica na Seção 3 deste enunciado.	**2.5%**
Reunião de Avaliação **Documentos a entregar:** - Relatório referente aos itens 14 a 18 (e seus subitens) especificados para a 4ª Reunião Periódica na Seção 3 deste enunciado.	**2.5%**

8 Avaliação do projeto final

A avaliação do projeto final totaliza 90% da classificação e observará o resultado final obtido na concretização dos seguintes componentes, já no sistema completo:

			Pesos
Design	1	Especificação API usando o padrão OpenAPI, em formato YAML ou JSON	10,0%
	2	Especificar requisitos NF - escalabilidade	1,0%
	3	Especificar requisitos NF - segurança	1,0%
	4	Especificar requisitos NF - tolerância a faltas	1,0%
	5	Design da arquitetura distribuída	10,0%
	5.1	Aplicação do conceito em relação aos load balancers	5%
	5.2	Aplicação do conceito em relação aos web servers	2,5%
	5.3	Aplicação do conceito em relação às bases de dados	2,5%
	6	GitHub - uso com branches individuais e releases	4,0%

Arquitetura distribuída e tolerância a faltas	7	Implementação da arquitetura distribuída (implementação de instâncias e configuração de redes)	12,0%
	7.1	Implementação dos LB	7%
	7.2	Implementação dos servers	2,5%
	7.3	Implementação das bases de dados	2,5%
	8	Configuração dos balanceadores de carga e mecanismos de escalabilidade	10,0%
	9	Configuração de mecanismos de tolerância a faltas e verificação de saúde	5,0%
	9.1	Teste de tolerância a faltas nos LB	3%
	9.2	Teste de tolerância a faltas nos webservers	1,0%
	9.3	Teste de tolerância a faltas nas bases de dados	1,0%
Segurança	10	Implementação de mecanismo de Autenticação e autorização (Auth0)	5,0%
	11	Canais seguros, DNS e configuração de firewall	10,0%
	11.1	Uso de TLS com certificado assinado por uma Autoridade Certificadora (AC)	4%
	11.2	Nome de domínio registado e associado ao IP estático	2%
	11.3	Configurações de firewall	4%
	12	Configurar uso de APIs externas	2,0%
	13	Gestão de credenciais e IPs na instalação	5,0%
Automação e testes	14	Automação e scripts para construção e lançamento da aplicação	2,00%
	15	Inclusão de testes unitários integrados ao processo de construção da aplicação (Github)	3,00%
	16	Testes de aceitação com a API	3,00%
	17	Testes de carga	3,00%
	18	Testes de vulnerabilidades	3,00%

9 Guiões e tutoriais

Um conjunto de guiões e tutoriais que podem usar como base para o desenvolvimento do projeto é apresentado abaixo:

Orientações gerais	https://12factor.net/
GitHub Hello World	https://guides.github.com/activities/hello-world/
GitHub Actions Hello World	https://lab.github.com/githubtraining/github-actions:-hello-world
OpenAPI	https://swagger.io/specification/
OpenAPI Tutorial	https://support.smartbear.com/swaggerhub/docs/tutorials/openapi-3-tutorial.html
Cliente para testes da API	https://www.postman.com
OpenAPI editor especificação	https://editor.swagger.io/
OpenAPI gerador da especificação (alternativa ao editor)	https://www.apimatic.io/
Load Balancer - HAProxy	http://cbonte.github.io/haproxy-dconv/2.5/intro.html e Guião "Balanceadores de Carga / HAProxy" da disciplina de PTR
Load Balancer - Nginx	https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/
Nginx - Docker container	https://hub.docker.com/_/nginx
Load Balancer redundante (obriga a nome de domínio)	https://www.nginx.com/resources/glossary/dns-load-balancing/
Monitorização	https://www.netdata.cloud
Segurança	https://owasp.org/
Top 10 OWASP	https://owasp.org/www-project-top-ten/

Top 10 OWASP - Recomendações	https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf.pdf
Autenticação	https://auth0.com/
Reserva de nome de domínio gratuito	Devem ser identificados os serviços que oferecem nomes gratuitamente no arranque do projeto de PTR.
Certificados SSL	https://letsencrypt.org/
Testes unitários	Python: https://docs.python.org/3/library/unittest.html, PHP: https://phpunit.de/
Testes de carga	https://locust.io/
Testes de vulnerabilidade	https://nmap.org/, https://www.zaproxy.org/, https://w3af.org/, https://sqlmap.org/

10 Procedimentos para a gestão do repositório

Devem ser seguidos os seguintes procedimentos:

- Criar projeto privado no GitHub;
- Editar configurações do projeto e adicionar como colaboradores os elementos do grupo e os docentes de PTR;
- Todos os elementos do grupo devem contribuir para o repositório. Nenhum elemento do grupo trabalha diretamente sobre o *main Branch*;
- Cada contribuição é adicionada a um *Branch* específico criado para o efeito. Quando a contribuição estiver preparada para ser avaliada pelos outros elementos é criado um *Pull Request*. Quando o *Pull Request* for aprovado o *Branch* é eliminado. Não misturar contribuições não relacionadas no mesmo *Branch*. Documentar cada *Branch* e *Pull Request*;
- Quando um *Commit* cria uma versão que está pronta para execução deve ser criada uma *Tag* (*Create a New Release*) que vai criar uma nova *Release* permitindo usá-la em qualquer altura para demonstrar esse estado do projeto.