Faculdade de Design, Tecnologia e Comunicação
Universidade Europeia

Proposta de Projeto de Sistemas Distribuídos

Sistema 2FA com aplicação de telemóvel

https://github.com/hugomeduarte/sistema2fa

20190843 - Hugo Duarte

20190972 - Luís Felipe Baptista

Descrição do Problema

Pretendemos através deste projeto implementar um serviço web simples tolerante a faltas e com suporte a autenticação de dois fatores com chaves de uso únicas temporais. Pretendemos ainda aplicar parte do código de servidor desenvolvido a um outro projeto de outra unidade curricular, de forma a fortalecer o processo de autenticação deste.

É esperado de qualquer serviço web moderno que o mesmo seja capaz de permanecer ativo a todo o tempo (ou a grande maioria do tempo), e que seja capaz de escalar para atender números grandes de clientes. Estes dois objetivos, tolerância a faltas e escalabilidade, apresentam muitas dificuldades para o criador de um sistema. Estes problemas podem ser atacados através da organização de componentes de hardware e software, e regras de comunicação entre as mesmas em um sistema distribuído. Pretendemos atingir estes objetivos em pequena escala para nosso projeto.

Mecanismos de autenticação são necessários para garantir a confidencialidade da informação em sistemas, especialmente se forem acessíveis pela web, e a prática comum da indústria é conduzir a autenticação através de senhas (algo que só indivíduos sabem). Se uma senha for comprometida, a confidencialidade dos dados do utilizador também estarão. Ao combinar senhas com chaves de uso único temporais (algo que só os indivíduos possuem, em um determinado momento), o potencial risco de quebra de confidencialidade dos dados no sistema é reduzido tremendamente[1]. Felizmente, a tendência é que cada vez mais sistemas usam dois ou mais fatores para autenticação[1].

Há um único caso de uso para o sistema que desenvolveremos, que é o acesso a um serviço web (que não será desenvolvido) através da autenticação dos utilizadores por dois fatores. Utilizadores terão de criar uma conta no serviço, descarregarem uma aplicação de autenticação simples nos seus telemóveis, e associarem a senha criada à aplicação de autenticação. Tendo feito isso, a aplicação de autenticação passará a mostrar a chave de uso único temporal (TOTP) para o serviço, e em subsequentes tentativas de acesso ao serviço, utilizadores terão de providenciar o e-mail e senha da conta, e a TOTP.

Descrição da Solução

Para oferecermos um serviço simples tolerante a faltas e seguro, criaremos um servidor para nossa aplicação com suporte a 2FA, que se comunicará com uma aplicação de autenticação simples para telemóvel, e que será replicado, assim como uma base de dados para armazenar dados de contas, com senhas cifradas. Haverão ainda reverse proxies entre os clientes e servidores, que servirão para balancear carga. Os servidores, bases de dados, e proxies serão replicados em máquinas virtuais.

Para a seguinte explicação assume-se que o cliente já possui uma conta criada no serviço. No momento que um cliente tentar se conectar a um servidor, um reverse proxy redirecionará o cliente para um servidor com menor carga. Depois que uma conexão for estabelecida, o cliente pedirá a página de login do serviço, e o servidor responderá. O cliente providenciará sua senha e e-mail, e se as credenciais estiverem corretas, prosseguirá para a verificação do TOTP. O servidor enviará a TOTP periódicamente para a aplicação de autenticação no dispositivo, e verificará a TOTP providenciada. Com a autenticação feita, o cliente poderá acessar ao serviço (que será uma tela em branco, com um texto parabenizando o cliente por ter se autenticado corretamente).

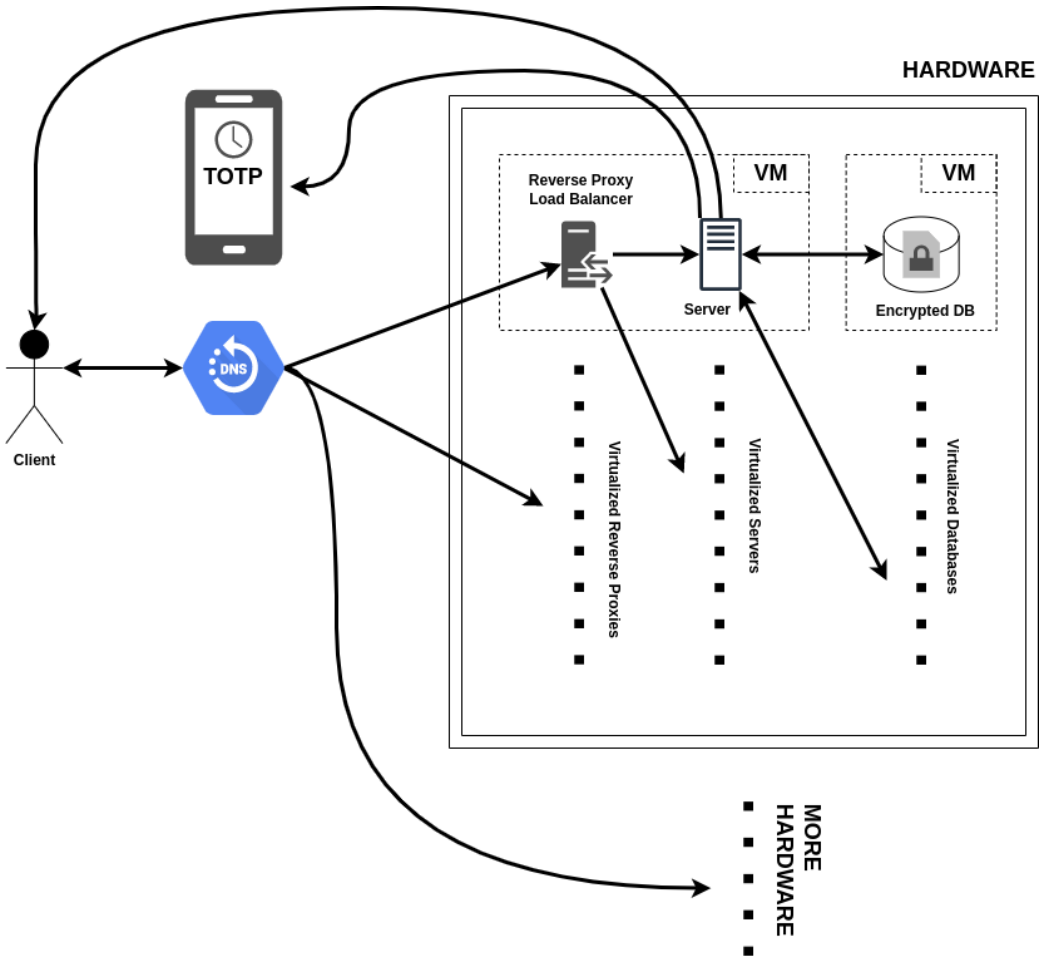


Figura 1 - Arquitetura do Caso Geral da Solução

O mecanismo mais simples e comum para atingir tolerância a faltas em um sistema é a replicação de recursos[2]. A replicação também possibilita a escalabilidade, uma vez que o balanceamento de carga entre várias máquinas de uma tarefa paralelizável, como o atendimento de pedidos de vários clientes em simultâneo, diminui o atraso de execução das tarefas. A replicação através da virtualização é uma forma econômica de se atingir tolerância a faltas e escalabilidade, e é uma técnica muito comum utilizada por serviços na Web.

Pretendemos replicar uma vez cada componente do sistema, de forma a provar o conceito mas manter o design e implementação do sistema simples. Isto é, pretendemos implementar o caso específico da arquitetura geral, ilustrado na Fig. 1, para dois reverse proxies, dois servidores, e duas bases de dados, todos virtualizados. As bases de dados existirão em uma VM cada, e os pares de reverse proxies e servidores também. A partilha de VMs dos servidores e proxies é feita para economizar recursos. Um servidor DNS resolverá os IPs dos reverse proxies, um de cada vez, pelo cliente. Os proxies farão o balanceamento de carga.

Será necessário garantir que a comunicação entre as componentes que passarem por redes seja cifrada, com TLS, com um certificado auto-assinado. As bases de dados estarão encriptadas[4]. Será também necessário garantir que acessos concorrenciais às bases de dados não levem a problemas, e que os estados das bases de dados estejam sincronizados.

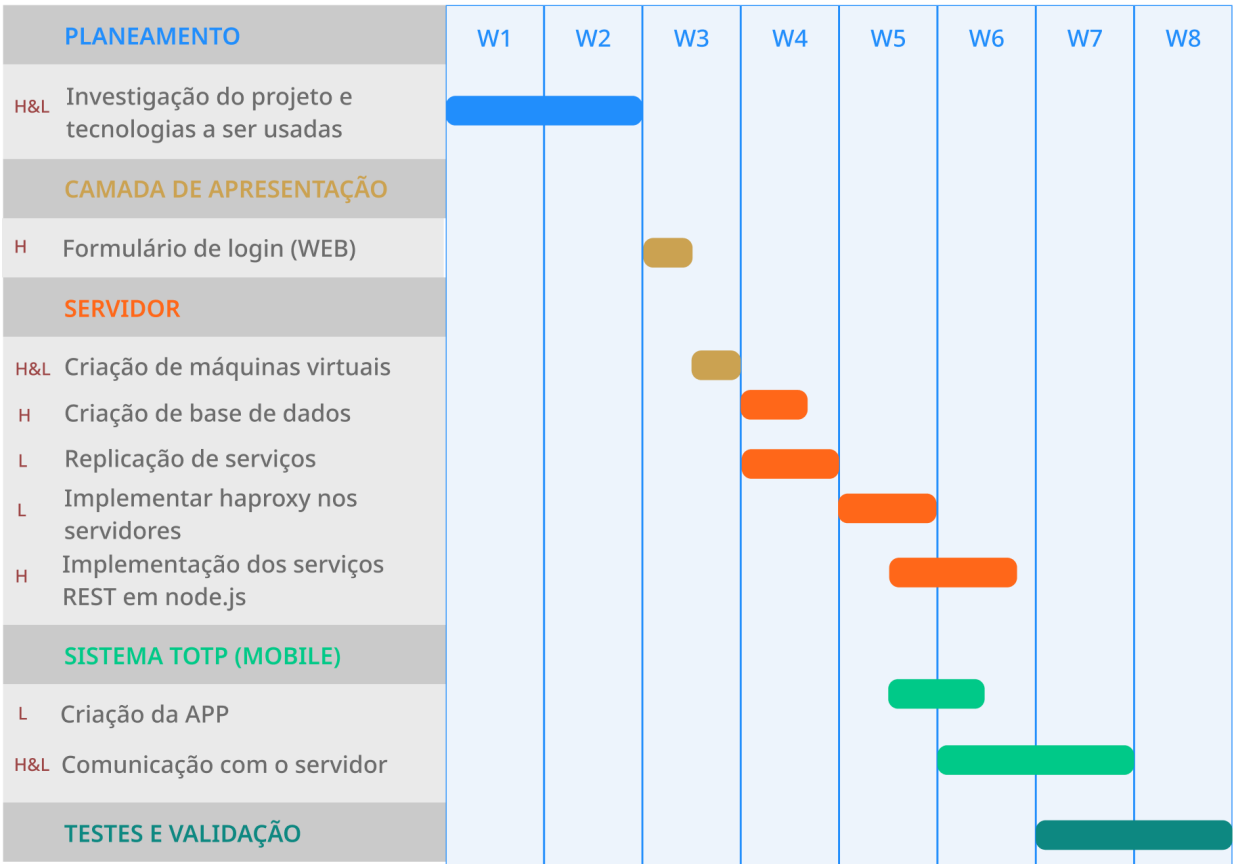
Tecnologias a utilizar

- **VirtualBox**: A virtualização será feita no virtualbox, no qual, será instalado o sistema operativo linux por ser um sistema mais leve, permitindo assim um melhor desempenho do mesmo.
- **HAProxy**: Conhecido como um reverse proxy para aplicações TCP e HTTP, permitirá um aumento de performance da aplicação, pois, distribuirá o workload do mesmo pelas diferentes replicações do servidor.
- **Node.js + Express.js**: Express.js é uma web framework para node.js. Será usada para a criação da aplicação web.
- **PostgreSQL**: Serviço de base de dados que será utilizado para armazenar a informação dos utilizadores e as suas chaves TOTP respetivas.
- **Bcrypt:** Algoritmo de criptografia hash para a encriptação da base de dados compatível com node.js.[5]
- **Flutter**: Ferramenta open source, criada pela Google para desenvolver aplicações multi-plataforma. Utilizaremos para desenvolver a aplicação de autenticação do telemóvel.

Planeamento

Estando numa fase muito inicial(fase de investigação do projeto), este diagrama de Gantt representa tópicos mais gerais de implementação do mesmo, não entrando assim em detalhes muito técnicos. No futuro atualizaremos o diagrama com esses mesmos detalhes e a sua respectiva distribuição de tarefas em cada fase ocorrida.

(H - Hugo, L - Luís, H&L - Hugo e Luís)



Referências

[1] *Information messages*. TechWeb RSS. (n.d.). Retrieved March 4, 2022, from https://www.bu.edu/tech/support/information-security/why-use-2fa/

[2] -, P. H. O., Por, -, & One, H. (2019, October 9). *Entenda sobre a replicação de dados*. Blog Host One. Retrieved March 4, 2022, from https://blog.hostone.com.br/replicacao-de-dados/

[3] Coates, M. (2014, June 4). Safely Storing User Passwords: Hashing vs. Encrypting. Dark Reading. https://www.darkreading.com/risk/safely-storing-user-passwords-hashing-vs-encrypting

[4] LaViska, C. (2017, February 8). *Hashing passwords with Node.js and bcrypt*. A Beautiful Site. Retrieved March 4, 2022, from

https://www.abeautifulsite.net/posts/hashing-passwords-with-nodejs-and-bcrypt