Licenciatura em Engenharia e Gestão de Sistemas de Informação

Universidade do Minho

Fundamentos dos Sistemas Distribuídos

2025/2026

Gestor de Parques de Estacionamento

Grupo 3 PL3

Afonso Leite

a108552

@uminho.pt

João Ribeiro

a108545

@uminho.pt

José Silva

a104070

@uminho.pt

Implementação

Arquitetura Geral

A Fase 4 introduz uma camada crítica de segurança no sistema distribuído desenvolvido nas fases anteriores, reforçando a comunicação HTTP existente com garantias explícitas de autenticidade e integridade. Enquanto as fases iniciais se centraram essencialmente na funcionalidade, na distribuição dos componentes e na interoperabilidade entre entidades, esta fase acrescenta mecanismos criptográficos que asseguram que as mensagens trocadas são efetivamente emitidas pelas entidades legítimas e que o seu conteúdo não foi alterado durante a transmissão.

A arquitetura de segurança adotada baseia-se numa Infraestrutura de Chave Pública simplificada, na qual o Gestor de Parques assume o papel de Autoridade de Certificação (CA). Este componente passa a ser o elemento central do modelo de confiança do sistema, sendo responsável pela emissão e validação de certificados digitais. O Cliente Web confia explicitamente na chave pública do Gestor e, por transitividade, passa a confiar em qualquer Parque de Estacionamento que possua um certificado válido emitido por este.

Do ponto de vista arquitetural, as principais alterações estruturais introduzidas na Fase 4 são as seguintes:

- **Gestor de Parques (Autoridade de Certificação):** para além das funcionalidades previamente existentes de registo e monitorização dos parques ativos, o Gestor passa a aceitar pedidos de registo que incluem chaves públicas RSA. Com base nessa informação, valida os pedidos recebidos e emite Certificados Digitais X.509, assinados com a sua chave privada, que associam a identidade do parque à respetiva chave pública.
- **Parques de Estacionamento (Emissores Seguros):** cada parque gera, no seu arranque, um par de chaves criptográficas RSA. A chave privada é mantida de forma confidencial e utilizada para assinar digitalmente todas as respostas produzidas pelos novos endpoints seguros, nomeadamente /secure/info e /secure/custo. A chave pública é enviada ao Gestor de Parques no momento do registo, permitindo a emissão do respetivo certificado digital.
- **Cliente Web (Validador):** o Cliente Web deixa de aceitar respostas de forma implícita. Sempre que recebe uma resposta proveniente de um Parque de Estacionamento, executa um processo de validação em duas etapas. Em primeiro lugar, verifica a autenticidade e validade do certificado do parque utilizando a chave pública do Gestor de Parques. Em segundo lugar, valida a assinatura digital da mensagem recebida, garantindo que o conteúdo não foi adulterado e que foi efetivamente produzido pelo parque correspondente.

Esta arquitetura mantém a compatibilidade com os componentes anteriores, como os simuladores de lugares (TCP), isolando a complexidade criptográfica na camada de comunicação externa (HTTP/REST). A segurança é assegurada através da biblioteca

cryptography, utilizando algoritmos robustos como SHA256 para *hashing* e esquemas de *padding* específicos (PKCS1v15 para certificados e PSS para mensagens) para prevenir vulnerabilidades comuns.

Estrutura dos Módulos

A implementação da Fase 4 preserva a organização modular do sistema, assente numa abordagem de componentes independentes com responsabilidades bem definidas. No entanto, foram introduzidas alterações significativas na lógica interna dos principais módulos, de forma a suportar a nova arquitetura de segurança baseada em Infraestrutura de Chave Pública (PKI). Adicionalmente, foi criado um módulo auxiliar dedicado à configuração e gestão das chaves criptográficas associadas à Autoridade de Certificação.

1. Parque de Estacionamento (parque.py)

Este módulo sofreu a maior evolução funcional. Deixou de atuar exclusivamente como um servidor de dados para assumir também o papel de entidade criptográfica ativa no sistema.

- **Gestão de Identidade**: no momento do arranque, o parque gera um par de chaves criptográficas RSA de 2048 bits e passa a gerir o ciclo de vida do seu certificado digital, necessário para a comunicação segura com os clientes.
- **Registo Seguro**: a função registar_no_gestor foi reestruturada para interagir com o endpoint seguro /parque_certificado, enviando a chave pública do parque e armazenando localmente o certificado digital devolvido pelo Gestor de Parques.
- **Assinatura Digital:** o módulo implementa mecanismos para assinar digitalmente as respostas enviadas aos clientes, recorrendo ao algoritmo de hashing SHA-256 e ao esquema de padding PSS, assegurando a integridade e a autenticidade das mensagens.
- **Novos Endpoints**: disponibiliza as rotas /secure/info e /secure/custo, que encapsulam a mensagem original juntamente com a assinatura e o certificado.

2. Cliente Web (cliente_web.py)

O Cliente Web assume, nesta fase, o papel de validador de confiança do sistema. A sua lógica interna foi significativamente expandida, passando a realizar verificações de segurança antes de qualquer processamento ou apresentação dos dados recebidos.

- **Motor de Validação**: Cliente Web integra a função validar_resposta_segura, responsável por executar a validação criptográfica em duas etapas. Numa primeira fase, é verificada a autenticidade do certificado X.509 do Parque de Estacionamento através da validação da sua assinatura com a chave pública do Gestor de Parques. Numa segunda fase, é validada a assinatura digital da mensagem recorrendo à chave pública do Parque, garantindo a integridade e autenticidade dos dados recebidos.
- **Gestão de Chaves**: o Cliente Web carrega e mantém em memória a chave pública do Gestor de Parques a partir do certificado X.509 fornecido pelos docentes (manager_cert.pem). Esta chave pública funciona como âncora de confiança (CA)

para validar certificados de Parques e, por transitividade, validar as assinaturas das mensagens recebidas.

Para garantir interoperabilidade num ambiente distribuído com múltiplos Parques de Estacionamento, o Cliente Web implementa uma validação tolerante a variações na serialização JSON das mensagens, testando diferentes representações possíveis antes de concluir a verificação da assinatura.

3. Módulos Auxiliares (lugar.py, protocolo.py e config.py)

Os restantes módulos do sistema mantiveram a sua estrutura original, evidenciando a robustez e a separação de responsabilidades da arquitetura adotada.

- **lugar.py:** a comunicação interna baseada em sockets TCP entre os sensores de lugar e o Parque de Estacionamento permanece inalterada. Esta comunicação ocorre num contexto controlado e fechado, não sendo necessária a introdução de mecanismos de criptografia assimétrica.
- **protocolo.py**: continua a ser responsável pela definição e serialização das mensagens trocadas via TCP, mantendo o isolamento entre a camada de sensores e a camada web segura.
- **config.py:** preserva a definição dos parâmetros de configuração globais do sistema, sem alterações significativas no âmbito da Fase 4.

Simulador de Lugares e Protocolo de Comunicação

Embora esta fase do projeto não tenha introduzido alterações diretas no simulador de lugares nem no protocolo TCP, ambos continuam essenciais para o bom funcionamento do sistema.

O simulador continua a criar os estados LIVRE e OCUPADO conforme as probabilidades e a transmiti-los ao parque via TCP.
O protocolo mantém as mensagens INIT, UPDATE, INFO e ERRO que são responsáveis pela comunicação estruturada entre os lugares e o Parque de Estacionamento.

A comunicação TCP interna entre os sensores (lugares) e o parque manteve-se isolada da camada HTTP pública, não sendo necessário aplicar a criptografia neste nível, visto que ocorre numa rede interna simulada.

Parque de Estacionamento e Gestor de Parques

1. Papel do Gestor de Parques

O Gestor de Parques sofreu uma evolução significativa, deixando de atuar exclusivamente como um serviço de registo e monitorização de endereços IP para assumir formalmente o papel de Autoridade de Certificação do sistema. Esta evolução materializa-se na introdução de um novo endpoint de registo seguro, acessível através do método POST

/parque_certificado. Através deste endpoint, os Parques de Estacionamento passam a enviar, para além dos seus dados de identificação, a respetiva chave pública criptográfica. Ao receber um pedido de registo contendo uma chave pública, o Gestor gera e assina digitalmente um certificado X.509, devolvendo-o, por sua vez, ao Parque.

2. Papel dos Parques de Estacionamento

Os Parques de Estacionamento passaram a assumir o papel de emissores seguros de informação, sendo responsáveis por produzir respostas autenticadas e íntegras para os clientes externos. Para esse efeito, foram introduzidos dois novos endpoints protegidos:

- GET /secure/info
- GET /secure/custo

Ao contrário dos endpoints definidos nas fases anteriores, estas interfaces não se limitam a devolver dados funcionais em texto simples. Cada resposta é agora estruturada como um objeto JSON composto por três elementos fundamentais:

1. **Mensagem**: contém os dados funcionais do parque, como lotação, número de lugares livres e informação tarifária.
2. **Certificado**: corresponde ao certificado digital X.509 do parque, codificado em UTF-8, utilizado para comprovar a identidade do emissor.
3. **Assinatura**: resulta da assinatura do hash SHA-256 da mensagem com a chave privada do parque, sendo codificada em cp437 para transporte no JSON.

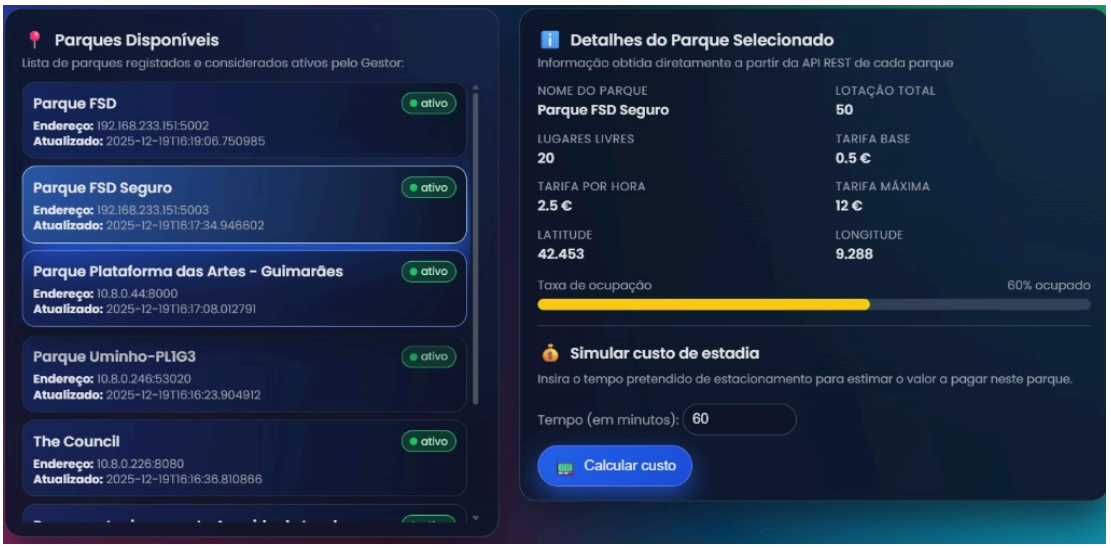

Figura 1: Interface do Cliente Web com os detalhes do “Parque FSD Seguro”, obtidos através do endpoint /secure/info após validação da assinatura digital e do certificado X.509.

Concorrência e sincronização

A introdução dos mecanismos de segurança na Fase 4 acrescentou novos desafios ao nível da concorrência, uma vez que o sistema passou a lidar com operações criptográficas, como RSA e SHA-256. Este fator reforçou a necessidade de um modelo de execução eficiente e devidamente sincronizado.

1. Concorrência no lado do Cliente Web (Flask)

O servidor Flask do Cliente Web continua a gerir vários pedidos em simultâneo, assumindo agora maiores responsabilidades. Cada resposta recebida de um Parque de Estacionamento implica o parsing do objeto JSON, a reconstrução do certificado digital X.509 e a verificação matemática da assinatura digital. Dado o custo computacional destas operações, o modelo multithreaded do Flask é fundamental para garantir que a validação de um pedido, como a simulação do custo de estacionamento, não bloqueia a navegação nem a atualização da informação apresentada a outros utilizadores.

2. Concorrência no Parque (Assinatura e Bloqueios)

O Parque de Estacionamento opera num ambiente de elevada concorrência interna, no qual coexistem vários fluxos de execução. Uma thread TCP recebe atualizações frequentes dos sensores de lugares, enquanto uma thread dedicada assegura o registo periódico e a renovação do certificado digital junto do Gestor de Parques. Paralelamente, múltiplas threads Flask respondem a pedidos externos através da API segura, implicando a geração de hashes e a assinatura de mensagens com a chave privada do parque. Para garantir a coerência dos dados, recorreu-se a mecanismos de sincronização, nomeadamente o threading.Lock, protegendo o acesso às estruturas partilhadas durante a geração e assinatura das respostas, evitando inconsistências que comprometam a integridade das mensagens.

3. Sincronização de Credenciais

Para além da sincronização dos dados funcionais, foi introduzida uma sincronização específica ao nível da segurança. O Parque de Estacionamento dispõe de uma thread dedicada à gestão do ciclo de vida do certificado digital, responsável pela renovação periódica do registo junto do Gestor. Este mecanismo garante a manutenção automática da relação de confiança, mesmo em cenários de reinício do sistema ou expiração de credenciais, sem necessidade de intervenção manual.

Considerações finais

A Fase 4 permitiu concluir o projeto com sucesso, transformando o sistema inicial num ambiente distribuído seguro. A implementação rigorosa dos requisitos definidos, nomeadamente a codificação das assinaturas em cp437 e a distinção entre o uso de padding PSS para mensagens e PKCS1v15 para validação de certificados demonstrou a capacidade de integrar mecanismos criptográficos avançados numa arquitetura REST.