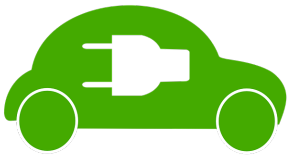
DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

LETI

Sistemas Distribuídos

1º Semestre – 2018/2019

Enunciado do Projeto: Sistema ECar



Introdução

O objetivo do projeto de Sistemas Distribuídos é desenvolver um sistema de gestão de carros elétricos partilhados, chamado ECar. O sistema deverá ser desenvolvido com *Web Services* na plataforma *Java*.

O sistema é suportado por uma rede de parques espalhados geograficamente pela cidade. Os utilizadores do sistema ECar podem localizar parques com carros disponíveis ou com lugares para devolução de carros.

1. Primeira parte: Sistema ECar

O sistema ECar é composto por uma rede de parques. Cada parque tem um conjunto de lugares, nas quais são colocados os veículos. Em cada momento, cada lugar pode estar livre ou ocupado por um carro. Inicialmente, um parque tem os lugares todos ocupados.

Além das coordenadas geográficas, cada parque tem outros atributos relevantes: a sua capacidade (número de lugares) e o prémio de entrega (bónus de pontos que um utilizador ganha ao devolver o carro nesse parque).

Cada utilizador tem uma conta no sistema, identificada por um endereço de *email* e protegida por uma palavra-chave[1]. A cada conta está associado um saldo. Inicialmente, cada utilizador começa com um saldo de 10 pontos.

Quando se aproxima de um parque com carros disponíveis, um utilizador pode alugar um desses carros. Ao alugar o veículo, o saldo do utilizador é subtraído em 1 ponto. Caso o uti-

[1] A proteção com senha apenas será adicionada ao trabalho na parte 3. Nas partes 1 e 2, o utilizador apenas precisa de se identificar com o seu endereço de *email*.

lizador tenha saldo positivo, não tenha nenhuma outro carro alugado nesse momento e exista pelo menos um carro disponível no parque em causa, o carro é disponibilizado ao utilizador.

Posteriormente, o utilizador deve devolver o carro num parque. Caso o parque de devolução tenha um prémio definido (diferente de 0, que é o valor por omissão), o saldo do utilizador recebe os pontos referentes a esse prémio.

A Figura 1 mostra uma visão global dos componentes da solução.

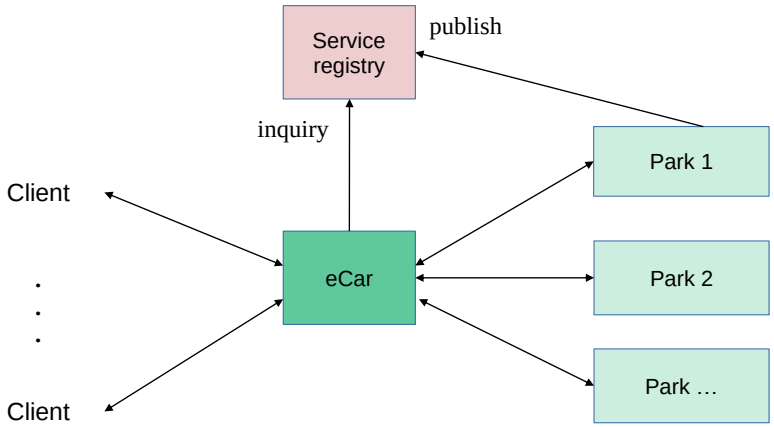


Figura 1: Arquitetura do projeto.

A componente central é o servidor ECar. Este servidor oferece um conjunto de serviços a dois tipos de utilizador: os utilizadores e o gestor da plataforma ECar. Os utilizadores são pessoas que adquiriram o direito a usar os carros disponíveis. Cada utilizador usa uma aplicação cliente. A aplicação invoca serviços do servidor ECar, permitindo ao utilizador consultar o saldo da sua conta, os parques e seu estado (número de carros e vagas disponíveis), alugar ou devolver carros. O gestor do ECar, por seu lado, tem acesso a funcionalidades de administração onde pode, por exemplo, consultar estatísticas sobre o uso do sistema,

Muitos dos serviços oferecidos pelo servidor ECar implicam invocar serviços oferecidos pelos parques espalhados pela cidade. Em cada parque existe um servidor que gere os dados desse parque. O objetivo do projeto é desenvolver os serviços que suportam o servidor principal e os dos parques. Fica fora do âmbito do projeto desenvolver a aplicação cliente .

1.1. Tecnologia

Todos os componentes do projeto serão implementados na linguagem de programação *Java*.
A invocação remota de serviços deve ser suportada por *Web Services* usando *JAX-WS*.

Os servidores de parque devem ser descobertos e localizados dinamicamente, por intermédio de um servidor UDDI. Cada servidor de parque é lançado autonomamente. Cabe a cada *Web Service* de parque, quando lançado, registar-se a si próprio no servidor UDDI bem conhecido, indicando o URL do servidor de parque. É no servidor UDDI que o servidor do ECar irá consultar a lista de parques para estabelecer contacto com os seus *Web Services*. Os nomes a usar para os serviços são: `CXX_ECar`, `CXX_Park1`, `CXX_Park2`, `CXX_Park3`, etc[2].

[2]O identificador do grupo tem o formato CXX, onde: C representa o campus (A para Alameda e T para Taguspark); XX representa o número do grupo de SD atribuído pelo Fénix. Por exemplo, o grupo T07 corresponde ao grupo 7 sediado no Taguspark.

1.2. Servidor ECar

De seguida descrevem-se as operações oferecidas pelo servidor ECar.

A operação `activateUser` ativa um novo utilizador no sistema, levando o servidor ECar a criar e inicializar o estado interno sobre este utilizador. Recebe o endereço de *email* do novo utilizador, que identifica a sua conta. Assume-se que o endereço de *email* segue o formato geral *utilizador@dominio*, em que utilizador e domínio são sequências de caracteres alfa-numéricos, com uma ou mais partes separadas por ponto "." e que não podem ser vazios.

A operação `getParkView` retorna os atributos, estatísticas e estado atual do parque passado como argumento. Os atributos retornados consistem nas coordenadas geográficas e capacidade. As estatísticas retornadas incluem: número acumulado de levantamentos e de entregas. O estado atual consiste no número atual de carros disponíveis e número atual de lugares livres.

A operação `getNearbyParks` lista os n parques (n dado como argumento) mais próximos das coordenadas indicadas como argumento. Os parques mais próximos são retornados em primeiro lugar. Para cada parque, é indicado o seu nome, as suas coordenadas, número de carros e os lugares disponíveis no momento.

As operações `rentCar` e `returnCar` procedem ao levantamento e entrega (respetivamente) de um carro no parque indicado em argumento, por parte do utilizador identificado em argumento.

A operação `getCredit` devolve o saldo atual do utilizador identificado pelo endereço de *email* passado como argumento.

As operações estão descritas de forma detalhada no seguinte contrato WSDL:

`http://disciplinas.tecnico.ulisboa.pt/leic-sod/2018-2019-sem1/labs/proj/ecar.1_0.wsdl`

1.3. Servidor de parque

Algumas das operações oferecidas pelo servidor do ECar precisam de invocar operações dos servidores de parque. De seguida descrevem-se essas operações.

As operações `getInfo` e `getStats` retornam informação e estatísticas sobre o parque.

As operações `rentCar` e `returnCar` procedem ao levantamento e entrega (respetivamente) de um carro no parque.

As operações estão descritas de forma detalhada no seguinte contrato WSDL:

`http://disciplinas.tecnico.ulisboa.pt/leic-sod/2018-2019-sem1/labs/proj/park.1_0.wsdl`

1.4. Operações auxiliares

Estas operações destinam-se a auxiliar os testes e não necessitam elas próprias de ser testadas exaustivamente. Por convenção, o seu nome começa por `test_`.

A operação `ping` responde com uma mensagem de diagnóstico não vazia que deve ilustrar o melhor possível o estado do sistema.

A operação `clear` limpa totalmente o estado do servidor invocado; ou seja, coloca-o no

estado inicial com os valores por omissão.

As operações `init...` permitem definir parâmetros de configuração do servidor.

1.5. Testes

Espera-se que cada projeto inclua testes que permitam cobrir todos os requisitos funcionais do enunciado.

Os *testes de integração* (IT) verificam o cumprimento do contrato de um Web Service através de invocações *remotas*. Devem usar JUnit para invocar todos os servidores remotos necessários (que se assume terem sido previamente lançados de forma correta).

Para os testes de integração deverão ativar **uma** instância do servidor ECar e pelo menos **três** parques.

Para o cenário de testes por omissão assume-se que o mapa da cidade tem uma quadrícula de 100x100 quadrados, com coordenadas de (0,0) até (99,99). Cada quadrado tem *lado* de 400 metros.

Para os testes, assumem-se os seguintes valores: o parque 1 está posicionado em (22,7), tem capacidade 6 e recompensa 2 pontos; o parque 2 está posicionado na posição (80,20), tem capacidade 12 e recompensa de 1 ponto; o parque 3 na posição (50,50), tem capacidade 20 e recompensa de 0 pontos.

2. Segunda Parte: Tolerância a Faltas

A tolerância a faltas do serviço ECar tem alguns aspetos críticos. Em especial, é importante que os saldos de cada utilizador (pontos gastos, pontos ganhos) não se percam nem sejam corrompidos, mesmo sabendo que o servidor poderá falhar ocasionalmente. Por essa razão pretende-se estender o serviço com suporte à replicação do servidor ECar.

Pretende-se seguir a abordagem de replicação passiva, implementando o protocolo *primary-backup* estudado nas aulas teóricas. O sistema replicado pode assumir que existe um conjunto estático de 2 réplicas. Deverá existir uma réplica do servidor ECar que funcionará como servidor secundário. O servidor principal será responsável por garantir que o estado do servidor secundário é mantido atualizado. O servidor secundário é responsável por detetar a falha do servidor principal e, nessa ocorrência, assumir o seu papel.

Como modelos de interação e faltas, deve assumir-se que:

- Os gestores de réplica podem falhar silenciosamente mas não arbitrariamente (i.e., não há falhas bizantinas);
- No máximo, existe uma falha em simultâneo;
- O servidor secundário não falha;
- O sistema é síncrono e a comunicação não omite mensagens.

É desejável que, tanto quanto possível, as opções tomadas privilegiem o desempenho da resposta ao cliente. No entanto, as otimizações não devem pôr em causa a correção do sistema replicado – mais precisamente, a garantia de *consistência sequencial* deve ser sempre preservada.

3. Terceira Parte: Segurança

A solução construída até agora não autentica os utilizadores do ECar, o que não é de todo recomendado. Permite, por exemplo, que um utilizador malicioso possa, através do seu cliente, invocar operações que afetem o saldo de um outro utilizador de forma ilegítima.

Na terceira parte do projeto pretende-se que essas limitações sejam ultrapassadas complementando o servidor ECar com o protocolo Kerberos (versão V5). Por simplicidade, deve ser implementada a variante simplificada do protocolo que é lecionada nas teóricas (sem servidores TGS autónomos, exigindo apenas um pedido-resposta para o cliente obter o *ticket* necessário para invocar o servidor).

O servidor de autenticação encontra-se já desenvolvido pelos docentes. Este servidor permite aos clientes do ECar pedirem uma nova sessão no ECar através de um web service SOAP com um WSDL e URL bem conhecido.

Este servidor conhece um conjunto de contas de utilizadores que se assumem pré-registadas. Os nomes de utilizadores e respetivas senhas secretas serão facultados aos grupos posteriormente. A chave secreta de cada utilizador u, usada pelo protocolo Kerberos, é dada por $resumo(senha(u), c)$, em que $hash$ é uma função de resumo criptográfica bem conhecida e c um número constante. O servidor de autenticação pré-conhece também as chaves secretas do servidor ECar de cada grupo de projeto, que também serão facultadas posteriormente aos grupos respetivos.

Após o cliente ter solicitado o início de nova sessão Kerberos e obtido os elementos respetivos (chave de sessão e *ticket* respetivo), pode invocar operações do servidor ECar de forma autenticada. Seguindo o protocolo, cada pedido ao servidor ECar deve transportar o *ticket* e autenticador anexados ao pedido, através do cabeçalho SOAP; por outro lado, a resposta a esse pedido deve incluir no cabeçalho a prova de frescura prevista pelo protocolo. Este procedimento deve ser seguido para qualquer invocação feita pelo cliente ao servidor ECar; está fora do âmbito do projeto autenticar as invocações entre o servidor ECar e os servidores de parque.

Antes de um cliente poder realizar qualquer operação, deve invocar a operação `activateUser` sobre o servidor ECar. Esta invocação serve para o servidor ECar passar a conhecer o utilizador já registado no servidor de autenticação.

O pedido e a resposta da comunicação entre o cliente e o ECar devem ser cifrados com a chave de sessão do Kerberos. Por simplificação, a integridade do pedido **não** necessita de ser garantida através de um MAC (*Message Authentication Code*).

A colocação dos elementos de segurança nas mensagens SOAP e o controlo de acessos devem ser feitos através de *JAX-WS Handlers*, que permitem intercetar a invocação de operações remotas no cliente e no servidor.

Como no projeto de SD não se desenvolve uma aplicação cliente, o início de sessão junto do servidor Kerberos deve ser feito pelos testes JUnit. Mais precisamente, uma bateria de testes deve começar por solicitar o início de nova sessão junto do servidor Kerberos (obtendo a chave de sessão e *ticket* respetivo) e, usando os elementos dessa sessão, executar as invocações remotas sobre o servidor ECar.

4. Avaliação

4.1. Colaboração e Entregas

O Git é um sistema de controlo de versões do código fonte que é uma grande ajuda para o trabalho em equipa. O seu uso é recomendado, mas *opcional*.

Caso pretendam usar Git, os estudantes devem atualizar a sua foto pessoal no GitHub, enviar os seus números, nomes e *usernames* por correio eletrónico para o professor dos laboratórios, com o assunto [GitHub for SD, please].

A cada parte do projeto a entregar estará associada uma *tag*. Cada grupo tem que marcar o código que representa cada entrega a realizar com uma *tag* específica – `SD_P1`, `SD_P2`, e `SD_P3` – antes da hora limite de entrega.

Será também possível entregar o trabalho através do Fénix.

4.2. Serviços externos

Alguns dos serviços necessários serão disponibilizados para utilização remota:

- UDDI: `http://uddi.sd.rnl.tecnico.ulisboa.pt:9090`
- Kerberos: `http://sec.sd.rnl.tecnico.ulisboa.pt:8888/kerbist`

4.3. Qualidade do código

A qualidade da estrutura base engloba os seguintes aspetos de avaliação (em todas as partes):

- Configuração (POMs e *Handlers*)
- Código legível (incluindo comentários relevantes)
- Tratamento de exceções adequado
- Sincronização correta

4.4. Primeira parte (P1)

A primeira parte vale 8 valores em 20, distribuídos da seguinte forma:

- Servidor Park
  - Qualidade da estrutura base (5%)
  - Implementação das operações (15%)
  - Testes de integração desenvolvidos pelo grupo (10%)
- UDDI e restante suporte para múltiplos parques (10%)
- Servidor ECar
  - Qualidade da estrutura base (15%)
  - Implementação das operações (25%)
  - Testes de integração desenvolvidos pelo grupo (20%)

A data limite de entrega é: *sexta-feira, 2 de novembro de 2018, 20:00*.

4.5. Segunda parte (P2)

A segunda parte vale 6 valores em 20, distribuídos da seguinte forma:

- Qualidade da estrutura base (20%)
- Replicação de dados no secundário (20%)
- Deteção de falta do primário (20%)
- Substituição do primário (20%)
- Demonstração (20%)

A data limite de entrega é: *sexta-feira, 23 de novembro de 2018, 20:00*.

4.6. Terceira parte (P3)

A terceira parte vale 6 valores em 20, distribuídos da seguinte forma:

- Qualidade da estrutura base (20%)
- Autenticação de utilizadores (30%)
- Cifra das mensagens de pedidos e respostas (30%)
- Demonstração (20%)

A data limite de entrega é: *sexta-feira, 7 de dezembro de 2018, 20:00*.

4.7. Discussão Final

Cada grupo deve preparar uma *demonstração* com os casos de utilização que demonstram as melhores funcionalidades do trabalho. Seguem-se sugestões para estruturar a demonstração, quer para a parte de tolerância a faltas, quer para a parte de segurança, .

Tolerância a faltas – Duração inferior a *5* minutos

- Caso *F1*: passos para demonstrar o funcionamento normal da replicação
- Caso *F2*: passos para demonstrar tolerância a falta

Segurança – Duração inferior a *5* minutos

- Caso *S1*: passos para demonstrar o funcionamento normal da segurança
- Caso *S2*: passos para demonstrar resistência a um ataque

Todos os alunos têm discussão final do projeto; nesta discussão qualquer aluno pode ter a sua nota obtida até então alterada. As notas das fases (P1 a P3) são indicativas e sujeitas a confirmação na discussão final na qual todo o software desenvolvido durante o semestre será tido em conta. As notas atribuídas são individuais.

4.8. Atualizações

Para as últimas novidades sobre o projeto, consultar a página Web regularmente:

`http://disciplinas.tecnico.ulisboa.pt/leic-sod/2018-2019-sem1/labs/`

O esclarecimento de dúvidas será realizado através do Piazza:

`https://piazza.com/tecnico.ulisboa.pt/fall2018/sd19_1`

Bom trabalho!