DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

LEIC-A, LEIC-T, LETI, MEIC-T, MEIC-A

Sistemas Distribuídos

2º Semestre – 2017/2018

# Enunciado do Projeto: Sistema Binas

## Introdução

O objetivo do projeto de Sistemas Distribuídos é desenvolver um sistema baseado em Web Services, implementados na plataforma Java, para a gestão de uma plataforma de bicicletas partilhadas chamada Binas.

À semelhança de sistemas como as GIRA (em Lisboa), a plataforma Binas consiste numa rede de estações de bicicletas partilhadas, distribuídas por uma cidade. Os utilizadores do sistema Binas, através de uma aplicação em *smartphone*, podem localizar estações próximas com bicicletas (ou vagas) disponíveis e, quando junto a uma estação, levantar uma bicicleta para posteriormente a devolver noutra estação.

## 1. Primeira parte: Sistema Binas

O Binas é composto por uma rede de estações, distribuídas por uma área geográfica. Cada estação tem um conjunto de docas, nas quais são colocadas as bicicletas partilhadas. Em cada momento, cada doca pode estar livre ou ocupada (por uma bicicleta). Inicialmente, uma estação tem todas as docas ocupadas (ou seja, tantas bicicletas disponíveis quanto o número de docas).

Além das coordenadas geográficas[1], cada estação tem outros atributos relevantes: a sua capacidade (número de docas) e o prémio de entrega (bónus de pontos que um utilizador ganha ao devolver bicicleta nessa estação). Estes atributos podem ser diferenciados entre as estações,

[1]Por simplificação assume-se que o mapa da cidade é uma quadrícula com coordenadas $x$ e $y$.

o que permite ao gestor da rede ajustar a capacidade de cada estação (adicionando/removendo docas para ajustar à procura) e encorajando os utilizadores a devolver bicicletas em estações que têm tendência a estar vazias (atribuindo um prémio).

Cada utilizador tem uma conta no sistema, identificada por um endereço de *email* e protegida por uma palavra-chave[2]. A cada conta está associado um saldo. Inicialmente, cada utilizador começa com um saldo de 10 pontos.

Quando se aproxima de uma estação com bicicletas disponíveis, um utilizador pode solicitar uma das bicicletas disponíveis nessa estação. Não existem identificadores individuais nas bicicletas. Ao levantar a bicicleta, o saldo do utilizador é subtraído em 1 ponto. Caso o utilizador tenha saldo positivo, não tenha nenhuma outra bicicleta levantada nesse momento e exista pelo menos uma bicicleta disponível na estação em causa, a bicicleta é destrancada e associada àquele utilizador.

Posteriormente, o utilizador deve devolver a bicicleta numa estação (a mesma ou outra qualquer) que tenha pelo menos uma doca livre. Caso a estação destino 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.

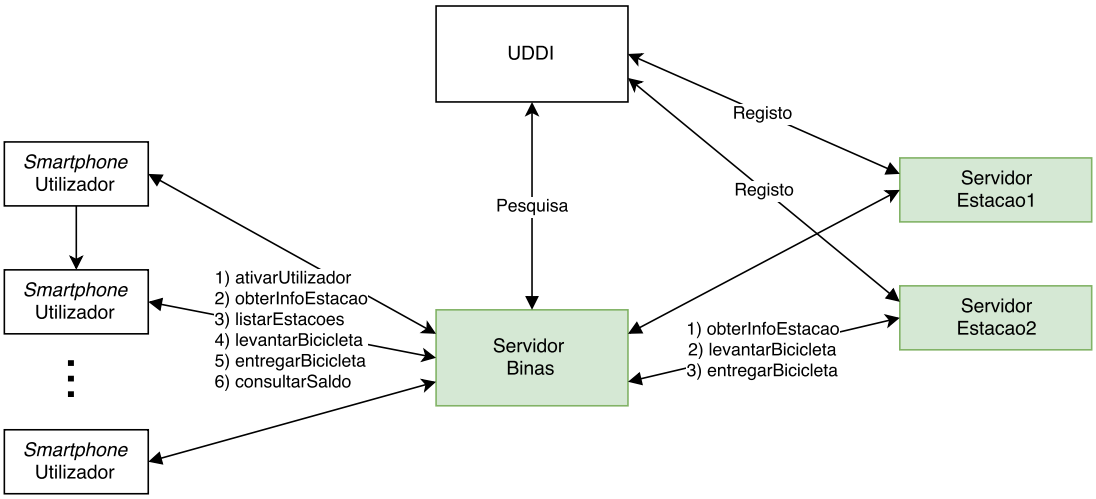


Figura 1: Arquitetura do projeto.

A componente central é o servidor Binas. Este servidor oferece um conjunto de serviços a dois tipos de utilizador: os utilizadores e o gestor da plataforma Binas. Os utilizadores são pessoas que adquiriram o direito a usar as bicicletas disponíveis. Cada utilizador usa uma aplicação cliente no seu *smartphone*. A aplicação invoca serviços do servidor Binas, permitindo ao utilizador consultar o saldo da sua conta, as estações disponíveis e seu estado (número de bicicletas e vagas disponíveis), levantar ou entregar bicicletas.

O gestor do Binas, por seu lado, tem acesso a funcionalidades de administração. Este também invoca serviços oferecidos pelo servidor Binas, permitindo ao gestor consultar estatísticas sobre o uso do sistema, como registar novas estações (assim como eliminar, redimensionar ou recolocar estações).

[2]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*.

Muitos dos serviços oferecidos pelo servidor do Binas implicam invocar serviços oferecidos pelas estações espalhadas pela cidade. Para tal, em cada estação existe um sistema embebido que aloja um servidor local e que gere as docas dessa estação. Em particular, o servidor de cada estação é capaz de trancar uma bicicleta que seja colocada numa doca (por um utilizador que devolve a bicicleta após a usar) e destrancar uma bicicleta estacionada numa doca (quando um utilizador pede para a levantar). O servidor local de cada estação mantém também um registo de levantamentos, entregas, entre outra informação de interesse estatístico. Todos estes serviços são invocados *indiretamente* (pelos utilizadores e pelo gestor do Binas) através do servidor Binas.

O objetivo do projeto é desenvolver os serviços que suportam os servidores principal e estações do Binas. Fica fora do âmbito do projeto desenvolver a aplicação móvel usada pelos utilizadores e a aplicação usada pelo gestor.

### 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 estação devem ser descobertos e localizados dinamicamente, por intermédio de um servidor UDDI. Cada servidor de estação é lançado autonomamente. Cabe a cada Web Service de estação, quando lançado, registar-se a si próprio no servidor UDDI bem conhecido, indicando o URL do servidor de estação. É no servidor UDDI que o servidor do Binas irá consultar a lista de estações para estabelecer contacto com os seus Web Services. Os nomes a usar para os serviços são: `CXX_Binas`, `CXX_Station1`, `CXX_Station2`, `CXX_Station3`, etc[3].

Não se exige nem será valorizado o armazenamento persistente do estado dos servidores. Isto aplica-se a todas as partes do projeto.

### 1.2. Servidor Binas

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

A operação `ativarUtilizador` ativa um novo utilizador no sistema, levando o servidor Binas 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 `obterInfoEstacao` retorna os atributos, estatísticas e estado atual da estação passada 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 bicicletas disponíveis e número atual de docas livres.

A operação `listarEstacoes` lista as $k$ estações ($k$ dado como argumento) mais próximas das coordenadas indicadas como argumento. As estações são retornadas em ordem crescente de distância às coordenadas indicadas. Para cada estação, é indicado o seu nome, as suas coordenadas, número bicicletas e as vagas disponíveis no momento.

[3]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 A22 corresponde ao grupo 22 sediado no campus Alameda; já o grupo T31 corresponde ao grupo 31 sediado no Taguspark.

As operações `levantarBicicleta` e `entregarBicicleta` procedem ao levantamento e entrega (respetivamente) de uma bicicleta na estação indicada em argumento, por parte do utilizador identificado em argumento.

A operação `consultarSaldo` 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/2017-2018/labs/proj/binas.1_0.wsdl`

### 1.3. Servidor de estação

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

A operação `obterInfoEstacao` retorna os atributos, estatísticas e estado atual da estação alvo, no mesmo formato devolvido pela operação homónima do servidor do Binas.

As operações `levantarBicicleta` e `entregarBicicleta` procedem ao levantamento e entrega (respetivamente) de uma bicicleta na estação.

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

`http://disciplinas.tecnico.ulisboa.pt/leic-sod/2017-2018/labs/proj/station.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 Binas e pelo menos três estações.

Para o cenário de testes por omissão deve-se assumir que o mapa da cidade tem uma quadrícula de 100x100 quadrados. A estação 1 está posicionada na posição (22,7), tem capacidade 6 e recompensa 2 pontos; a estação 2 está posicionada na posição (80,20), tem capacidade

12 e recompensa de 1 ponto; a estação 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 Binas tem alguns aspetos críticos. Em especial, é importante que as alterações aos saldos de cada utilizador (pontos gastos, pontos ganhos) não se percam nem sejam corrompidas, mesmo sabendo que o servidor Binas poderá falhar ocasionalmente. Por essa razão pretende-se estender o serviço com suporte à replicação dos saldos das contas de utilizador, o que dará maior disponibilidade às operações que manipulem esses mesmos saldos.

A solução planeada passa por aproveitar a existência dos servidores de estação para neles manter uma réplica dos saldos de conta dos utilizadores do Binas. Mais precisamente, cada servidor de estação passa a ser também gestor de réplica, oferecendo 2 operações adicionais ao servidor Binas: `lerSaldo` e `escreverSaldo`. A primeira recebe o identificador do utilizador como argumento e a segunda recebe o identificador de utilizador e o novo valor do saldo (entre outros argumentos que podem ser definidos pelo grupo).

Assim sendo, sempre que o servidor Binas precise de consultar ou alterar o saldo de um utilizador (nomeadamente, nas operações `consultarSaldo`, `levantarBicicleta` e `entregarBicicleta`), este poderá recorrer ao conjunto de gestores de réplica para aceder ao saldo do utilizador de forma tolerante a faltas.

Pretende-se seguir a abordagem de replicação ativa, implementando o protocolo *quorum consensus* estudado nas aulas teóricas para coordenar as leituras e escritas nas réplicas. O sistema replicado pode assumir que existe um conjunto estático de $N$ gestores de réplica (ou seja, servidores de estação); que não muda enquanto o sistema estiver em execução.

É desejável que, tanto quanto possível, as opções tomadas privilegiem o desempenho da resposta ao cliente. Recomenda-se que cada grupo considere variantes do protocolo *quorum consensus* original que, tendo em conta o cenário específico deste projeto, permitam um melhor desempenho do sistema. O uso de tecnologias como *invocação assíncrona* de Web Services pode ser especialmente útil para alcançar esses objetivos. No entanto, as optimizações/simplificações não devem pôr em causa a correção do sistema replicado – mais precisamente, a garantia de consistência sequencial deve ser sempre preservada.

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 minoria de gestores de réplica em falha em simultâneo;
- O sistema é assíncrono e a comunicação pode omitir mensagens (apesar do projeto usar HTTP como transporte, deve assumir-se que outros protocolos de menor fiabilidade podem ser usados).

## 3. Terceira Parte: Segurança

A solução construída até agora não autentica os utilizadores do Binas, 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 Binas 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 Binas pedirem uma nova sessão no Binas 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 Binas 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 Binas de forma autenticada. Seguindo o protocolo, cada pedido ao servidor Binas 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 Binas; está fora do âmbito do projeto autenticar as invocações entre o servidor Binas e os servidores de estação.

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

O pedido e a resposta podem seguir em claro, ou seja, não se exige que pedido/resposta sejam cifrados; mas espera-se que a integridade do pedido seja garantida através de um MAC (*Message Authentication Code*) usando a chave de sessão do Kerberos.

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 Binas.

# 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. Toda a partilha de código para trabalho deve ser feita através do GitHub.

A atualização do repositório deve ser feita com regularidade, correspondendo à distribuição de trabalho entre os alunos e às várias etapas de desenvolvimento. Cada elemento do grupo deve atualizar o repositório do seu grupo à medida que vai concluindo as várias tarefas que lhe foram atribuídas.

As entregas do projeto serão feitas também através do repositório GitHub[4]. O repositório de cada grupo estará disponível em: `https://github.com/tecnico-distsys/CXX-Binas/`

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.

### 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/kerby`

### 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:

- Estação
  - Qualidade da estrutura base (5%)
  - Implementação das operações (15%)
  - Testes de integração desenvolvidos pelo grupo (10%)
- UDDI e restante suporte para múltiplas estações (10%)
- Binas
  - Qualidade da estrutura base Binas (15%)
  - Implementação das operações Binas (25%)
  - Testes de integração Binas desenvolvidos pelo grupo (20%)

A data limite de entrega é: *sexta-feira, 13 de abril de 2018, 23:59*.

[4]Para grupos que não usam GitHub para trabalhar, podem recorrer a uma entrega via Fénix. A entrega via Fénix sobrepõe-se a uma entrega via GitHub.

### 4.5. Segunda parte (P2)

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

- Qualidade da estrutura base (20%)
- Replicação ativa (60%)
- Relatório e demonstração (20%)

A data limite de entrega é: *quarta-feira, 2 de maio de 2018, 23:59*.

### 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 (20%)
- Controlo de acessos (20%)
- Integridade de pedidos (20%)
- Relatório e demonstração (20%)

A data limite de entrega é: *sexta-feira, 18 de maio de 2018, 23:59*.

### 4.7. Relatório

O *relatório* deve ser entregue juntamente com o código, na pasta `doc/` na raiz do projeto. O ficheiro deve chamar-se `CXX-relatorio-tolfaltas/seguranca.pdf`, para a parte 2 e 3, respetivamente. O documento deve conter:

- 1 folha de rosto:
  - Identificador do grupo em formato CXX;
  - URL do repositório no GitHub;
  - Fotos, números e nomes dos membros do grupo (ordenados por número de aluno crescente, da esquerda para a direita).
- 2 páginas de conteúdo:
  - Figura da solução de segurança/tolerância a faltas (deve ocupar 1/2 página);
  - Descrição da figura e breve explicação da solução;
  - No caso da *tolerância a faltas*, deve ser detalhada a troca de mensagens do protocolo implementado;
  - No caso da *segurança*, deve ser detalhado o protocolo e os conteúdos das mensagens SOAP (pedido e resposta), com legenda e breve explicação.

### 4.8. Demonstração Final

Cada grupo deve preparar um *guião de demonstração* com casos de utilização que demonstram as melhores funcionalidades do trabalho. Os guiões não têm formato pré-definido e devem ser incluídos na pasta `doc/` na raiz do projeto em formato PDF. Os ficheiros devem chamar-se `CXX-guiao-tolfaltas/seguranca.pdf`, para a parte 2 e 3, respetivamente. O guião deve incluir instruções de instalação e configuração, que devem começar com a obtenção do código no repositório Git, devem dar indicações de compilação e de como proceder para fazer a demonstração preparada.

Seguem-se sugestões sobre como estruturar o guião, 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

### 4.9. Discussão Final

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.

### 4.10. Atualizações

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

`http://disciplinas.tecnico.ulisboa.pt/leic-sod/2017-2018/labs/`

O esclarecimento de dúvidas será realizado através do Piazza:

`https://piazza.com/tecnico.ulisboa.pt/spring2018/sd18`

Bom trabalho!