**Universidade Lusófona de Humanidades e Tecnologias**

**Licenciatura em Engenharia Informática**

**Computação Distribuída**

**Projeto: Serviço Biclas**

1. **Introdução**

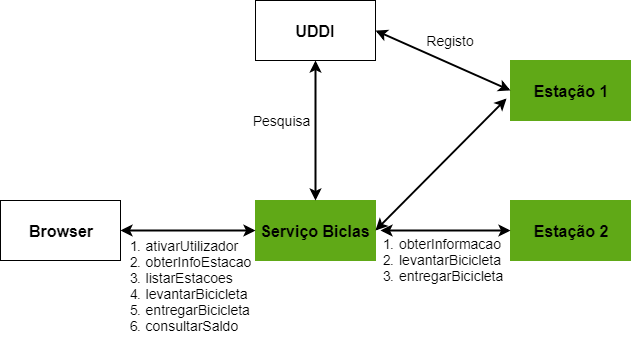
O objetivo do projeto é desenvolver um sistema baseado em WebServices implementados na plataforma Java para a gestão de uma plataforma de bicicletas partilhadas. À semelhança do sistema GIRA (em Lisboa), o serviço Biclas consiste numa rede de estações de bicicletas partilhadas, distribuídas por uma cidade. Os utilizadores acedem ao serviço Biclas através de um browser, onde 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 Biclas**

O Biclas 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, 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, o que permite ao gestor da rede ajustar a capacidade de cada estação (adicionando/removendo docas para ajustar à procura) e encorajar 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 (opcional). 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. Ao levantar a bicicleta, ao saldo do utilizador é subtraído 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 Serviço Biclas

Este servidor oferece um conjunto de serviços a dois tipos de utilizador: os utilizadores e o gestor da plataforma Biclas. Os utilizadores são pessoas que adquiriram o direito a usar as bicicletas disponíveis. Cada utilizador usa uma aplicação cliente no seu browser. A aplicação invoca serviços do servidor Biclas, 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 Biclas, por seu lado, tem acesso a funcionalidades de administração. Este também invoca serviços oferecidos pelo servidor Biclas, permitindo ao gestor consultar estatísticas sobre o uso do sistema, como registar novas estações (assim como eliminar, redimensionar ou recolocar estações).

Muitos dos serviços oferecidos pelo servidor do Biclas 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 outras informações de interesse estatístico. Todos estes serviços são invocados indiretamente (pelos utilizadores e pelo gestor do Biclas) através do servidor Biclas.

O objetivo do projeto é desenvolver os serviços que suportam o servidor principal e as estações do Biclas.

* 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. O servidor de estação deve ser descoberto e localizado 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. No servidor UDDI, o serviço Biclas irá consultar a lista de estações para estabelecer contacto com os seus Web Services. Os nomes a usar para os serviços são: GXXBiclas, GXXStation1, GXXStation2, GXXStation3, etc, onde XX é o número do grupo. Não se exige nem será valorizado o armazenamento persistente do estado dos servidores.

* 1. **Servidor Biclas**

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

A operação **ativarUtilizador** ativa um novo utilizador no sistema, levando o servidor Biclas a criar e inicializar o estado interno sobre este utilizador. Recebe o endereço de email do novo utilizador que identifica a 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 de bicicletas e as vagas disponíveis no momento.

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: **biclas\_1\_0.wsdl**.

**1.3. Servidor de estação**

Algumas das operações oferecidas pelo servidor do Biclas 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 Biclas.

As operações **levantarBicicleta** e **entregarBicicleta** procedem ao levantamento e entrega de uma bicicleta na estação.

As operações estão descritas de forma detalhada no seguinte contrato WSDL: **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**<qualquer coisa> permitem definir parâmetros de configuração do servidor.

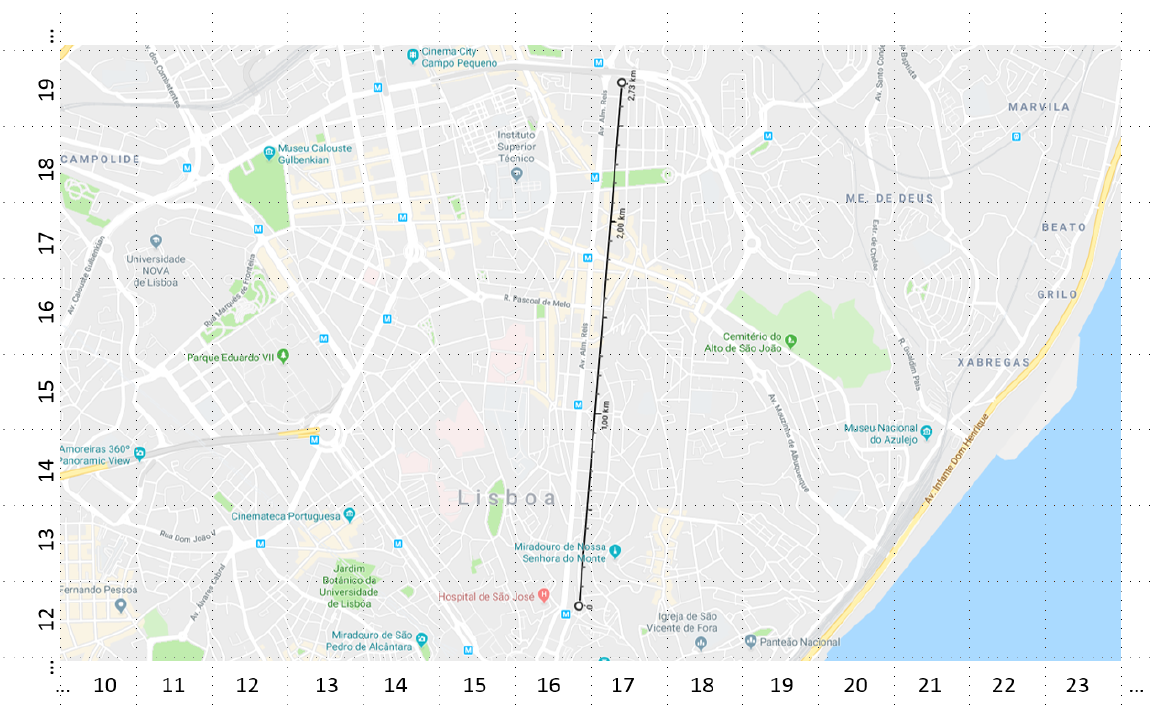
**1.5. Testes**

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 Biclas e pelo menos **três** estações.

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. A Figura 2 mostra uma parte do mapa.



**Figura 2:** Excerto do mapa de Lisboa dividido em quadrículas (meramente ilustrativo).

Para os testes, assumem-se os seguintes valores:

1. a estação 1 está posicionada na posição (22,7), tem capacidade 6 e recompensa 2 pontos;
2. a estação 2 está posicionada na posição (80,20), tem capacidade 12 e recompensa de 1 ponto;
3. 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 Biclas 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 corrompidos, mesmo sabendo que o servidor Biclas 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. Mais precisamente, cada servidor de estação passa a ser também gestor de réplica, oferecendo 2 operações adicionais ao servidor Biclas: **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 Biclas 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 um **protocolo simples de quórum** (VER) descrito no enunciado para coordenar as leituras e escritas nas réplicas. O sistema deve alcançar esses objetivos. No entanto, as otimizações/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:

1. Os gestores de réplica podem falhar silenciosamente;
2. No máximo, existe uma minoria de gestores de réplica em falha em simultâneo;
3. 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).
   1. **Protocolo de quórum**

Quórum é o número mínimo de votos que uma transação distribuída deve obter para poder realizar uma operação. Uma técnica baseada em quórum é implementada para impor uma operação consistente em um sistema distribuído.

No serviço Biclas, a cada leitura do saldo é invocado a operação **lerSaldo** em cada estação que irá devolver a informação juntamento com um timestamp da operação. O saldo correto do utilizador será obtido numa maioria de valores iguais. Por exemplo, assumindo que a conta do utilizador foi corrompida dentro de limites, pelo menos uma maioria de estações irá devolver um valor de saldo igual. É esse valor de saldo que irá ser apresentado ao utilizador. Se não se obter uma maioria de valor de saldo igual, a conta do utilizador está corrompida e irá ser apresentado o valor 0. Se uma minoria de estações tiver o saldo errado, esse valor será corrigido pelo valor correto.

A cada escrita do saldo (**escreverSaldo**), é enviado o mesmo saldo a todas as estações. Assume-se que não é preciso haver um sistema de votação nas escritas.

**3. Terceira Parte: Segurança (Projeto de 2ª Época)**

A solução construída até agora não autentica os utilizadores do Biclas, 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 Biclas com o protocolo **Kerberos** (versão V5). Por simplicidade, deve ser implementada a variante simplificada do protocolo(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 Biclas pedirem uma nova sessão no Biclas 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 posteriormente aos grupos.**­**

A chave secreta de cada utilizador usada pelo protocolo Kerberos, é dada por ***hash*(senha(username))**, em que hash é uma função de resumo criptográfica bem conhecida e com um número constante. O servidor de autenticação pré-conhece também as chaves secretas do servidor Biclas 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 Biclas de forma autenticada. Seguindo o protocolo, cada pedido ao servidor Biclas 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 Biclas; está fora do âmbito do projeto autenticar as invocações entre o servidor Biclas e os servidores de estação.

Antes de um cliente poder realizar qualquer operação, deve invocar a operação **ativarUtilizador** sobre o servidor Biclas. Esta invocação serve para o servidor Biclas 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 CD 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 Biclas.

**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. O repositório de cada grupo estará disponível em: <https://<GIT URL>/GXX-Biclas/>

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 – CD P1, CD P2, e CD P3

– antes da hora limite de entrega.

**4.2. Serviços externos**

Alguns dos serviços necessários serão disponibilizados para utilização remota:

1. UDDI: http://<URL>:9090
2. Kerberos: http://<URL>:8888/kerby

Os ficheiros WSDL estão disponibilizados em: https://<URL>

**4.3. Qualidade do código**

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

1. Configuração (POMs e Handlers)

Código legível (incluindo comentários relevantes):

1. Tratamento de exceções adequado
2. Sincronização correta

**4.4. Primeira parte (P1)**

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

**Estação**

1. Qualidade da estrutura base (5%)
2. Implementação das operações (15%)
3. Testes de integração desenvolvidos pelo grupo (10%)
4. UDDI e restante suporte para múltiplas estações (10%)

**Biclas**

1. Qualidade da estrutura base Biclas (15%)
2. Implementação das operações Biclas (25%)
3. Testes de integração Biclas desenvolvidos pelo grupo (20%)

A data limite de entrega é: domingo, DD de MM de 2018, 23:59. Será descontado 1 ponto por cada dia passado após o prazo.

**4.5. Segunda parte (P2)**

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

1. Qualidade da estrutura base (20%)
2. Replicação ativa (60%)
3. Relatório e demonstração (20%)

A data limite de entrega é: domingo, DD de MM de 2018, 23:59. Será descontado 1 ponto por cada dia passado após o prazo.

**4.6. Terceira parte (P3)**

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

1. Qualidade da estrutura base (20%)
2. Autenticação de utilizadores (20%)
3. Controlo de acessos (20%)
4. Integridade de pedidos (20%)
5. Relatório e demonstração (20%)

A data limite de entrega é: domingo, DD de MM de 2019, 23:59. Será descontado 1 ponto por cada dia passado após o prazo.

**4.7. Relatório**

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

1. 1 folha de rosto:
2. Identificador do grupo em formato GXX;
3. URL do repositório no GitHub;
4. Fotos, números e nomes dos membros do grupo (ordenados por número de aluno crescente, da esquerda para a direita).
5. 2 páginas de conteúdo:
6. Figura da solução de segurança/tolerância a faltas (deve ocupar 1/2 página);
7. Descrição da figura e breve explicação da solução;
8. No caso da tolerância a faltas, deve ser detalhada a troca de mensagens do protocolo implementado;
9. 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 **GXX-guiao-biclas.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 da cadeira regularmente.

O esclarecimento de dúvidas será realizado presencialmente nas aulas.

**10. Referências**

W3C - Web Services Architecture

http://www.w3.org/TR/ws-arch/