



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE
COIMBRA



departamento
de engenharia informática
1995 – 2020

Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Informática

Mestrado em Engenharia Informática
Integração de Sistemas
Projeto 3, 2020/2021, 1º Semestre

Docente

Pedro Nuno San-Bento Furtado
(PNF@DELUC.PT)

Grupo: 23

David Silva de Paiva – nº 2020178529 - PL2

Ricardo David Da Silva Briceño – nº 2020173503 – PL2

Registo de Trabalhos

Lista de funcionalidades/objetivos implementados

Funcionalidades/Objetivos	Implementado/Não implementado	Responsável	Esforço (horas)
Primeira implementação de uma arquitetura P2P	-	Ambos	12
User Node	Implementado	Ricardo	8
Novo registo no sistema	Implementado	Ricardo	1
Login no sistema	Implementado	Ricardo	1
Listagem de todos os títulos de publicações do sistema	Implementado	Ricardo	1
Obter publicação detalhada pelo título	Implementado	Ricardo	2
Adicionar uma nova publicação	Implementado	Ricardo	1
Atualizar uma publicação existente	Implementado	Ricardo	1
Remover uma publicação existe	Implementado	Ricardo	1
System Node	Implementado	David	13.5
Listar todos os utilizadores	Implementado	David	1/4
Listar todas as tarefas pendentes	Implementado	David	1/4
Aprovar/Rejeitar Registos	Implementado	David	1
Aprovar/rejeitar nova publicação	Implementado	David	2
Aprovar/rejeitar editar publicação	Implementado	David	2
Aprovar/rejeitar remover publicação	Implementado	David	2
Desativar/ativar um utilizador	Implementado	David	1
Mostrar informação detalhada de uma publicação	Implementado	David	1/4
Assegurar que só existe um administrador	Implementado	David	3/4
Outros aspetos e bug fixing	-	David	4
Relatório	Implementado	Ambos	8

Tabela 1 - Funcionalidades e Objetivos implementados no Projeto 3

Autoavaliação individual e global do projeto

O aluno, David Paiva, autoavalia o desempenho do grupo com uma nota de dezanove valores - numa escala de zero a vinte. Relativamente a uma autoavaliação individual, avalia o seu desempenho e o do colega, Ricardo Briceño, com dezanove valores para ambos.

O aluno, Ricardo Briceño, autoavalia o desempenho do grupo com uma nota de dezanove valores - numa escala de zero a vinte. Relativamente a uma autoavaliação individual, avalia o seu desempenho com quinze valores e o do colega, David Paiva, com dezanove valores.

Índice de Figuras

Figura 1 - Arquitetura do segundo projeto	1
Figura 2 - Criação de um durableConsumer que não irá perder nenhuma notificação	6
Figura 3 - Cabeçalho do Message Driven Bean: MyMsg	7
Figura 4 - Cabeçalho da classe UserEntity e respectivas propriedades~	8
Figura 5 - Modelo de dados utilizado para armazenar pedidos de adição/atualização/remoção de publicações	9

Índice de Tabelas

Tabela 1 - Funcionalidades e Objetivos implementados no Projeto 3	i
---	---

Índice

Registo de Trabalhos	i
Lista de funcionalidades/objetivos implementados	i
Autoavaliação individual e global do projeto	i
Índice de Figuras	iii
Índice de Tabelas	iii
1. Enquadramento do projeto	1
1.1 Objetivo geral	1
1.2 Arquitetura e funcionalidades	1
1.3 Estrutura do documento	2
2. User Node – Aplicação dos investigadores	5
2.1 Registo de novos utilizadores	5
2.2 Login – Acesso a aplicação dos investigadores	5
2.3 Listar todos os títulos das publicações existentes	5
2.4 Detalhes de uma publicação específica	5
2.5 Funcionalidades que exigem a aprovação do Administrador	6
2.5.1 Adicionar uma nova publicação	6
2.5.2 Atualizar informação de uma publicação existente	6
2.5.3 Remover uma publicação existente	6
2.6 Outros aspetos	6
3. System Node – Message Driven Bean (MyMsg)	7
3.1 Atendimento de pedidos	7
3.2 Persistência dos pedidos	8
4. Aplicação Administrator	11
4.1 AdministratorSessionBean	11
4.2 Desativação/Ativação de um utilizador	11
4.3 Notificação aos clientes das alterações	11
4.4 Garantia de um só administrador em execução	11
5. Considerações finais	13

Esta página foi deixada propositadamente em branco.

1. Enquadramento do projeto

Nesta secção são explicados, de uma forma concisa, os objetivos chave do projeto. Para além disso, é feito um breve enquadramento da arquitetura e funcionalidades da aplicação desenvolvida.

1.1 Objetivo geral

O segundo projeto, da unidade curricular de Integração de Sistemas, consiste na implementação de uma aplicação empresarial que suporta a interação de aplicações recorrendo ao sistema de mensagens Java Message Service (JMS). Para além disso foram utilizados alguns recursos do trabalho prático anterior, como é o caso do modelo de dados.

1.2 Arquitetura e funcionalidades

O terceiro projeto é composto por um conjunto de aplicações independentes que trocam informação entre si, e são controladas por dois tipos de utilizadores, os investigadores – User Node - e um administrador que é único no sistema - Administrator. A Figura 1 apresenta a arquitetura do projeto desenvolvido.

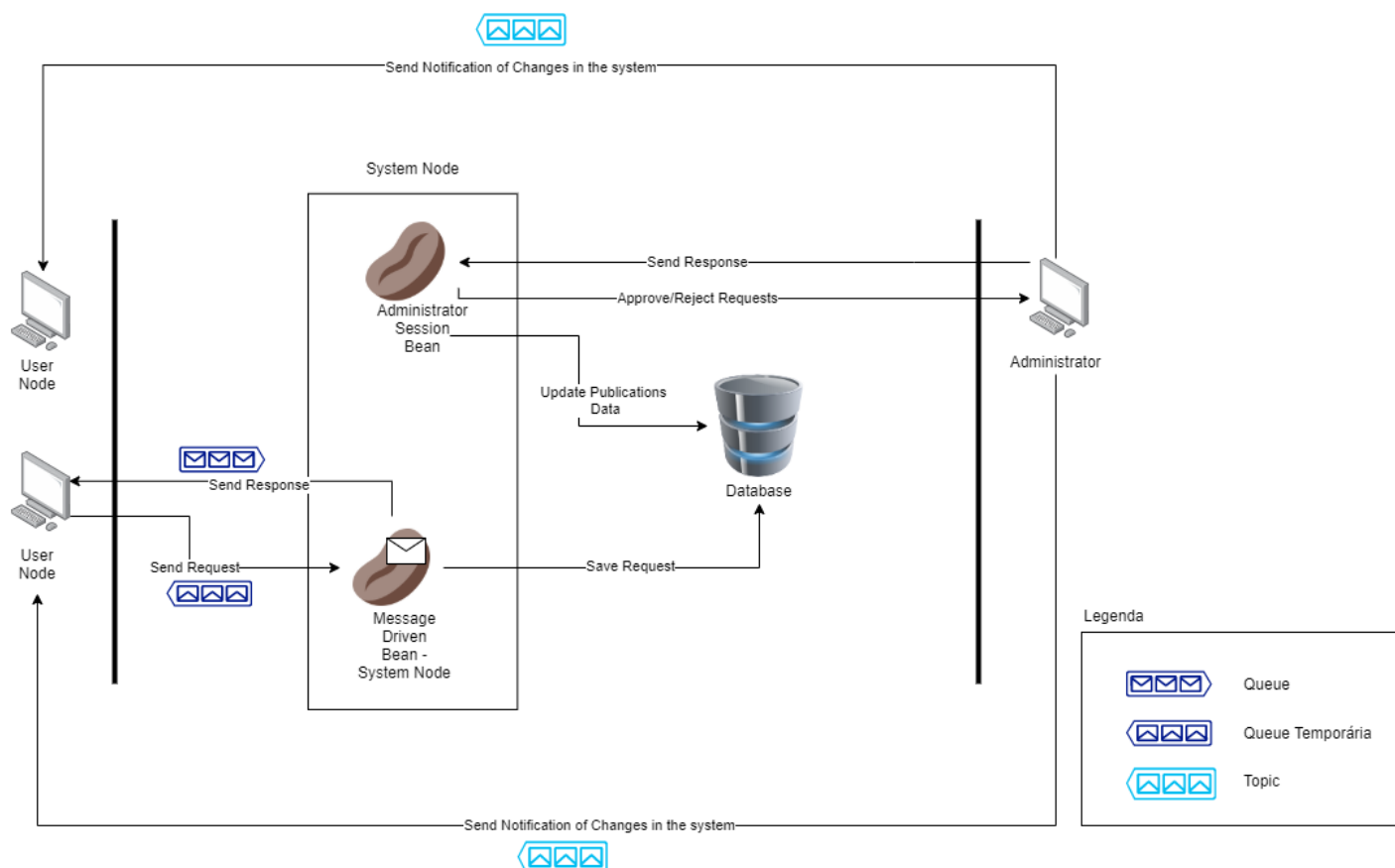


Figura 1 - Arquitetura do segundo projeto

A arquitetura do sistema sofreu algumas alterações no decorrer do desenvolvimento do projeto. Numa primeira fase, tinha-se desenhado uma arquitetura na qual as aplicações System Node e Administrador estavam integradas numa simples aplicação Java. Já a aplicação cliente era uma outra aplicação Java que enviava/recebia mensagens para/de o System Node. Esta arquitetura apresentava um grande problema de dependência, uma vez que era necessário que ambas as aplicações estivessem online para que fosse possível realizar o conjunto de operações descritas no enunciado.

Com isso em mente, foi adotada uma nova arquitetura que é apresentada acima. O System Node é composto por duas partes fundamentais:

- Um message driven bean responsável por receber os pedidos das aplicações UserNode e os guardar na base de dados para que não sejam perdidos e duplicados, mantendo assim uma consistência dos dados da aplicação.
- Um session bean responsável por realizar toda a lógica de negócio da aplicação Administrador, que é descrita mais à frente.

O User Node (aplicação do cliente), por sua vez, é uma aplicação Java simples na qual os utilizadores registados conseguem realizar pedidos que serão atendidos diretamente pelo message driven bean.

No que toca à comunicação entre as aplicações. os pedidos dos clientes são enviados através de uma queue para o message driven bean e, após serem atendidos, é enviada uma resposta através de uma queue temporária criada por cada cliente. Relativamente às notificações, estas são enviadas para um Topic, pelo administrador, sempre que é adicionada/editada/removida uma publicação, e, consequentemente, são recebidas por cada cliente e apresentadas na consola. Por fim, quando um utilizador realizar o primeiro login, após o registo ter sido aprovado pelo administrador, é lhe enviada uma notificação a avisar de que o registo foi aprovado.

1.3 Estrutura do documento

Para além deste enquadramento, este documento é composto pelos seguintes capítulos:

- Capítulo 2 – User Node

- Capítulo 3 – System Node – Message Driven Bean
- Capítulo 4 – Aplicação de Administração
- Capítulo 5 – Considerações Finais, onde é feito um balanço dos objetivos alcançados, assim como as competências adquiridas com a realização do projeto.

Esta página foi deixada propositadamente em branco.

2. User Node – Aplicação dos investigadores

Nesta secção, é apresentada a aplicação dos investigadores na qual pode haver múltiplos utilizadores a interagir simultaneamente com o sistema. Esta aplicação permite realizar todas as operações que foram propostas no trabalho.

2.1 Registo de novos utilizadores

É apresentada uma interface de consola que permite realizar o registo de novos utilizadores no sistema inserindo um username e uma password. O registo não é automático e precisa de ser aprovado pelo administrador. Após a aprovação ser concedida é guardada na base de dados uma indicação de que quando o utilizador realizar login, é necessário notificá-lo que o registo foi aprovado. Com isso em mente, no momento do primeiro login o utilizador recebe uma notificação de que o registo foi aprovado.

2.2 Login – Acesso a aplicação dos investigadores

Caso o utilizador já se tenha registado no sistema – e este tenha sido aprovado pelo administrador – poderá então aceder a todas as funcionalidades da aplicação. Caso contrário, terá de esperar pela aprovação do administrador.

No caso de o utilizador ter sido desativado pelo administrador também não conseguirá realizar o login e, portanto, não terá acesso às funcionalidades da aplicação.

2.3 Listar todos os títulos das publicações existentes

O cliente pede para que o sistema liste todos os títulos das publicações existentes. O seu pedido é enviado para uma queue que é acedida pelo message driven bean. Este responde com a lista de publicações existentes na base de dados.

2.4 Detalhes de uma publicação específica

Foi desenvolvida uma funcionalidade para exibir informação detalhada de uma publicação específica, seguindo a mesma lógica da funcionalidade anterior, mas, com a particularidade de se pedir ao utilizador o título da publicação a qual deseja ver em detalhe. O pedido e a resposta são processados da mesma forma que a funcionalidade anterior.

2.5 Funcionalidades que exigem a aprovação do Administrador

As funcionalidades desenvolvidas e apresentadas abaixo, exigem uma revisão e a devida aprovação do Administrador. Como já foi referido, estes pedidos são armazenados na base de dados para que possam ser aprovadas pelo administrador.

2.5.1 Adicionar uma nova publicação

Para adicionar uma nova publicação são pedidas, ao cliente, informações relevantes sobre a mesma. Esta funcionalidade permite associar um ou mais investigadores à publicação.

2.5.2 Atualizar informação de uma publicação existente

Para atualizar a informação de uma publicação já existente são solicitadas, ao cliente, informações referentes ao número de leituras, recomendações e citações.

2.5.3 Remover uma publicação existente

Para remover uma publicação são listadas todas as publicações e é pedido ao utilizador que escolha a que pretende remover do sistema.

2.6 Outros aspetos

No caso de um utilizador ficar offline, este não irá perder as notificações que, entretanto, foram enviadas para o Topic. Para isso foi apenas criado um *durableConsumer* e é cada cliente identifica-se com o um *id*, que neste caso é o username – como mostra a Figura 2.



Figura 2 - Criação de um *durableConsumer* que não irá perder nenhuma notificação

3. System Node – Message Driven Bean (MyMsg)

Nesta secção, é apresentado a aplicação responsável por receber os pedidos dos clientes e os armazenar numa base de dados de modo a que não sejam perdidos e para que, posteriormente, possam ser analisados por um administrador.

3.1 Atendimento de pedidos

Esta aplicação implementa um message driven bean (MyMsg) – Figura 3 - que, depois de *deployed* no servidor Wildfly, comporta-se como um JMS Listener, estendendo a interface *MessageListener* e, conseqüentemente, implementa o método *onMessage* que é responsável por tratar os diferentes pedidos dos clientes.



```
@MessageDriven(name = "MyMsg", activationConfig = {
    @ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Queue"),
    @ActivationConfigProperty(propertyName = "destination", propertyValue = "queue/playQueue") })
public class MyMsg implements MessageListener {
    ...
}
```

Figura 3 - Cabeçalho do Message Driven Bean: MyMsg

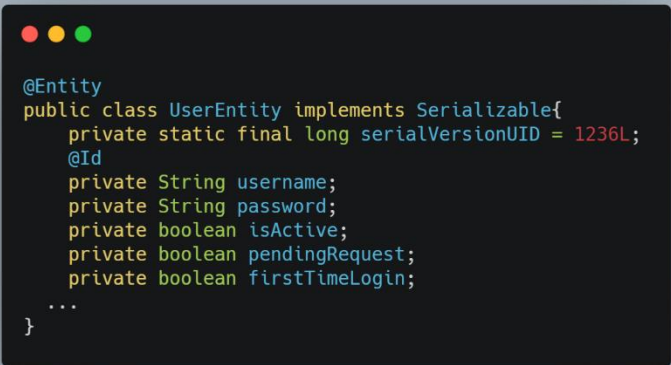
Nos pedidos de consulta de dados, relativos às publicações do sistema, o message driven bean consulta na base de dados as informações solicitadas e envia-a através de uma file de mensagens (queue) temporária criada pelo cliente que efetuou o pedido. Já quando são recebidos pedidos de registo ou pedidos que envolvem a alteração de informação guardada na base de dados, o message driven bean persiste essas mensagens numa base de dados – para que não sejam perdidas – e envia uma mensagem de confirmação ao cliente que efetuou o pedido. A referência para o EntityManager – responsável por realizar as operações na base de dados – é injetado através do container.

Foi vantajoso a utilização de um message driven bean uma vez que, assim, é possível atender clientes de forma assíncrona. Isto acontece porque depois de *deployed* no servidor, o bean está sempre à escuta de pedidos e consegue atender vários clientes simultaneamente.

3.2 Persistência dos pedidos

Os pedidos de registo de um novo utilizador e de alteração dos dados das publicações, que chegam ao message driven bean, são armazenados na base de dados para posteriormente serem aprovador/rejeitados por um administrador. Deste modo, é assegurado que não são aceites pedidos que possam colocar em questão a integridade e coerência dos dados do sistema.

Os pedidos de registo são adicionados à base de dados caso ainda não exista nenhum username igual ao do pedido. Deste modo, antes da gravação na base de dados, a propriedade *pendingRequest* é colocada com o valor “true” de forma a indicar que este utilizador ainda não se pode autenticar e aguarda confirmação do administrador. Para saber se é a primeira vez que o utilizador efetua o login, e desse modo notificá-lo de que o registo foi aceite pelo administrador, a flag “*firstTimeLogin*” guarda o valor “true” nessa situação e logo após o primeiro login passa a tomar o valor “false”. Na Figura 4 é apresentada o cabeçalho da classe *UserEntity* e as respetivas propriedades. Esta classe é mapeada para a base de dados através de *Java Persistence API*.



```
@Entity
public class UserEntity implements Serializable{
    private static final long serialVersionUID = 1236L;
    @Id
    private String username;
    private String password;
    private boolean isActive;
    private boolean pendingRequest;
    private boolean firstTimeLogin;
    ...
}
```

Figura 4 - Cabeçalho da classe *UserEntity* e respetivas propriedades~

Os pedidos que visam adicionar, editar e remover uma publicação são armazenados na base de dados com recurso a duas novas classes que, posteriormente, são mapeadas para duas novas tabelas na base de dados – Figura 5. A propriedade *actionToDo*, da tabela *PublicationRequest*, guarda um valor numérico que identifica o tipo de ação que é necessário realizar com a publicação em questão, podendo tomar os seguintes valores:

- 5 -> Adicionar publicação ao sistema.
- 6 -> Atualizar os dados da publicação do sistema.
- 7 -> Eliminar a publicação do sistema.

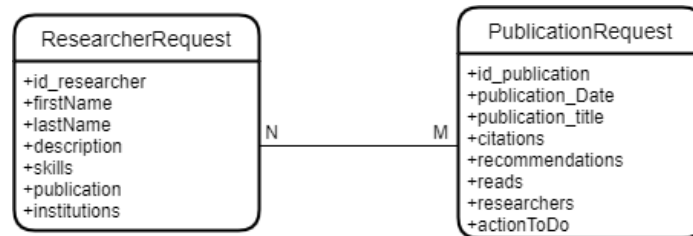


Figura 5 - Modelo de dados utilizado para armazenar pedidos de adição/atualização/remoção de publicações

Esta página foi deixada propositadamente em branco.

4. Aplicação Administrator

Nesta secção, é apresentada a aplicação responsável pela parte da administração, podendo ser executadas as seguintes operações:

- Listar todos os utilizadores;
- Listar todas os pedidos pendentes (registos, adição/atualização/remoção de publicações);
- Aprovação/Rejeição de pedidos pendentes;
- Desativação/Ativação de um utilizador;
- Listar todos os títulos de publicações no sistema;
- Mostrar informação detalhada de uma publicação;

4.1 AdministratorSessionBean

O session bean, `AdministratorSessionBean`, encapsula toda a lógica de negócio da aplicação do Administrator, nomeadamente todas as operações de consulta e alteração dos dados relativos a utilizadores e publicações – listadas acima.

4.2 Desativação/Ativação de um utilizador

A desativação/ativação de um utilizador é feita com recurso a uma *flag*, guardada na base de dados, que indica se o utilizador está ativo ou não. No momento do login é feita a verificação dessa *flag*.

4.3 Notificação aos clientes das alterações

Quando o administrador aprova um pedido de adição/atualização/remoção de uma publicação na base de dados, notifica todos os clientes do ocorrido através do envio de uma mensagem para um *Topic*. Todos os clientes estão registados como *listeners* e quando recebem a notificação, apresentam-na de imediato na consola.

4.4 Garantia de um só administrador em execução

Uma das restrições do enunciado é de que apenas poderia existir uma instância em execução da aplicação administrativa. Isto foi facilmente implementado através de um log na base de dados sempre que a aplicação administrator inicia. Basicamente, quando verifica se existe uma linha na tabela *AdminLog* e, caso exista, termina a execução alertando o administrador que já existe uma aplicação em execução. No caso de se verificar que não existe nenhuma linha na base de dados, é escrita uma linha e o

administrador pode então realizar as operações desejadas na aplicação. Quando o administrador desejar terminar a aplicação, a linha é eliminada da base de dados.

Esta é uma abordagem simples e apresenta um problema. Ou seja, se por acaso a aplicação administrador terminar abruptamente a execução, a linha de log não será eliminada da base de dados e, consequentemente, em futuras tentativas de execução da aplicação, o administrador receberá o alerta de que já existe uma aplicação em execução quando isso não é verdade. Devido à falta de tempo não foi implementado nenhum mecanismo tolerante a este tipo de falhas, no entanto no próximo parágrafo está descrita uma possível solução.

Uma solução possível para o problema apresentado passa por criar uma queue que de cinco em cinco segundos, por exemplo, recebe uma mensagem e tem um intervalo de tempo para responder com um *“i am alive”*. No caso de não ser recebida a resposta a linha de log era eliminada da base de dados. Esta é uma abordagem muito comum e relativamente fácil de implementar. Esta mensagem poderia ser enviada, por exemplo, pelo message driven bean que está instalado no servidor Wildfly. Deste modo era garantida a tolerância a falhas.

5. Considerações finais

Dado por terminado o projeto é necessário fazer um balanço dos objetivos alcançados, assim como as competências adquiridas.

De uma forma geral, todas as funcionalidades solicitadas no enunciado foram implementadas com sucesso. Com isto em mente, é importante realçar que os objetivos de aprendizagem foram alcançados. Nomeadamente, foram apreendidas um conjunto de técnicas e ferramentas no que toca ao desenvolvimento de aplicações que têm a necessidade de comunicar entre si e realizar troca de informação recorrendo a *Java Message Service*.

Consideramos, ainda, que as nossas soluções podem não estar da forma mais eficiente possível. Contudo, tentamos procurar sempre soluções ajustadas aquilo que nos é exigido e esperado no âmbito da disciplina de Integração de Sistemas. Para tal, tentamos desenvolver uma arquitetura que numa primeira fase pareceu-nos a mais correta, mas, reestruturamos a arquitetura com orientação do docente e concluímos que a arquitetura atual é mais eficiente e mais completa que a anterior.

Para além disso, é importante realçar alguns detalhes que numa aplicação real deveriam ser revistos:

- As passwords estão guardas na base de dados sem qualquer encriptação;
- Tolerância a falha da aplicação administrador.