**Documento de entrega.**

# Objetivos deste documento

Este documento formaliza a entrega bem a configuração de ambiente necessária para rodar o projeto.

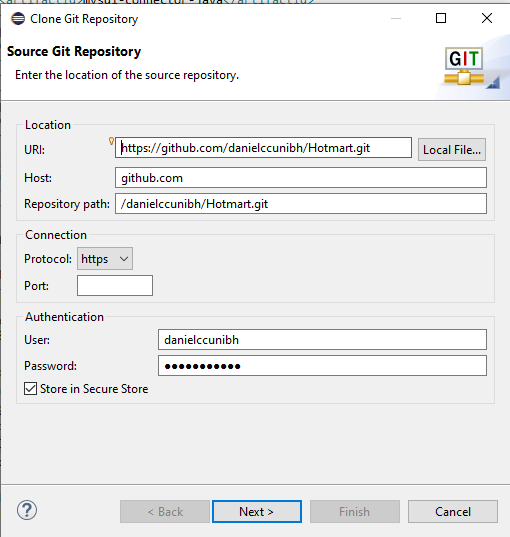
**Configuração do Ambiente**

# Softwares necessários:

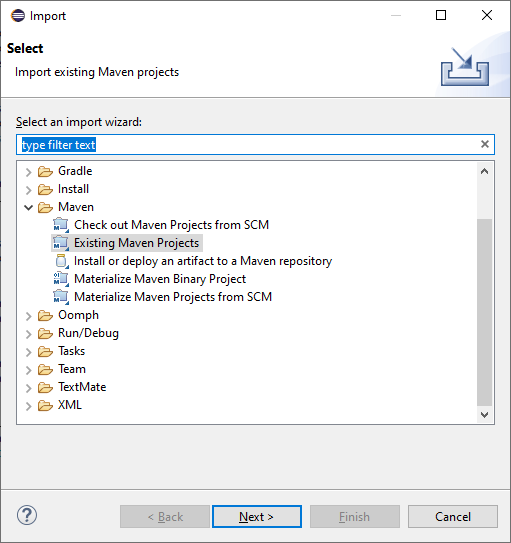
* Eclipse (última versão) <https://www.eclipse.org/downloads/packages/>
* Mysql (última versão) <https://dev.mysql.com/downloads/workbench/>
* Hazelcast (hazelcast-4.0.3) https://download.hazelcast.com/download.jsp?version=hazelcast-4.0.3&p=

# Etapas de instalação:

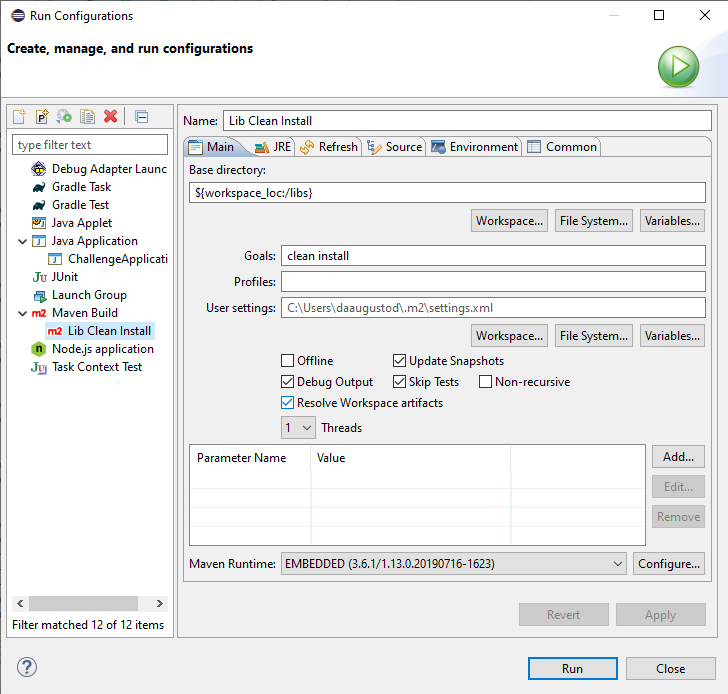
* Instale o eclispe com suas opções default.
* Faça a instalação da biblioteca lombok (<https://projectlombok.org/setup/eclipse>)
* Clone o repositório do github (<https://github.com/danielccunibh/Hotmart/tree/master>) no eclipse conforme imagem



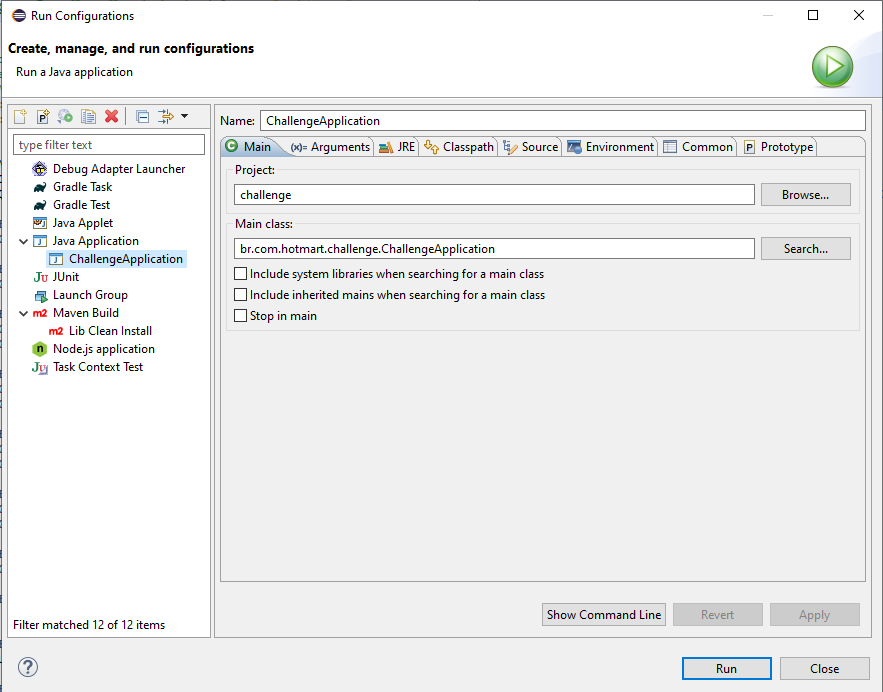
* Após clonar o repositório, baixe a branch master.
* Importe os projetos (lib e challenge) como projeto maven informando o diretório onde os projetos foram baixados.



* Após importação, crie o clean install do maven para o projeto lib e clique em run.



* Após a conclusão do build, clique com botão direito no projeto challenge, maven, update Project, marque todos os projetos e clique em ok.
* Instale o workbench e crie o banco de dados com o arquivo DDL.sql que se encontra no projeto no pacote “libs\src\main\java\br\com\hotmart\libs\sql”.
* O usuário e senha para o banco de dados foi configurado como root/admin no arquivo application.properties caso necessário alterar.
* Com a criação do banco de dados, rode o arquivo DML.sql para gerar a carga inicial do banco que se encontra na mesma pasta da linha anterior.
* Vá ao diretório aonde salvou o hazelcast, descompacte, acesse o diretório bin e execute o arquivo start.bat (Windows) ou start.sh (Linux).
* Inicie a aplicação challenge conforme imagem:



* Caso seja necessário, o modelo de dados se encontra no pacote “br.com.hotmart.libs.model” do projeto lib.
* Usuário e senha da aplicação para acessar as rotinas de PUT, DELETE e POST são admin/123456
* Documentação da api disponível em http://localhost:8080/swagger-ui.html#/.

# Questões pendentes

|  |  |  |
| --- | --- | --- |
| Pendencia | Responsável | Previsão |
| Conteinerização do projeto - Docker | Daniel Augusto | N/A |
| Testes automatizados | Daniel Augusto | N/A |
| Versionamento de Script (flyway) | Daniel Augusto | N/A |
| Teste unitários | Daniel Augusto | N/A |
| Auditoria de serviços | Daniel Augusto | N/A |

As questões acima ficaram pendentes devido ao tempo x tamanho do desafio x proficiência técnica. Os itens acima, caso fosse implementado, seriam feitos com as tecnologias:

Teste automatizado – Selenium

Teste unitário = Junit4

Conteinerização – Docker

Versionamento de Script – flyway

Auditoria de serviços – não consegui encontrar uma ferramenta que atendesse esse padrão de auditoria. A ideia para contornar essa deficiência seria incluir logs através do Logger (org.slf4j.Logger) e configurar o ENVERS ao projeto para que as alterações de tabela fosse auditadas. No caso do ENVERS, houve uma tentativa de aplicar, sendo modelado as tabelas e incluindo as dependências no pom do projeto, bem como algumas anotações as entitys. Infelizmente não houve prazo hábil para concluir esse recurso ficando pendente para uma próxima etapa. Os Testes unitários também não foram entregues para não comprometer o tempo do teste.

**Arquitetura proposta**

