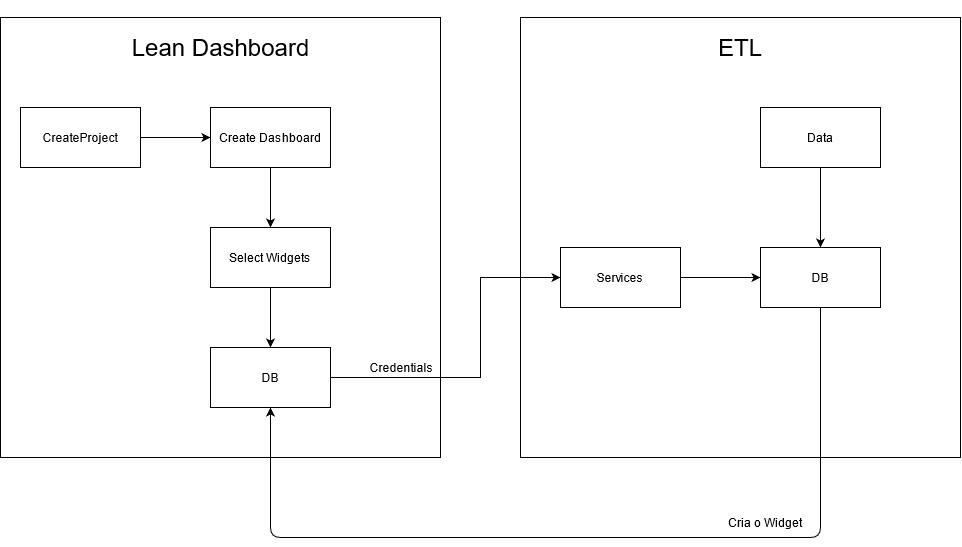
**Reunião 20/05/21**

**Fluxo de dados/Estrutura do código**

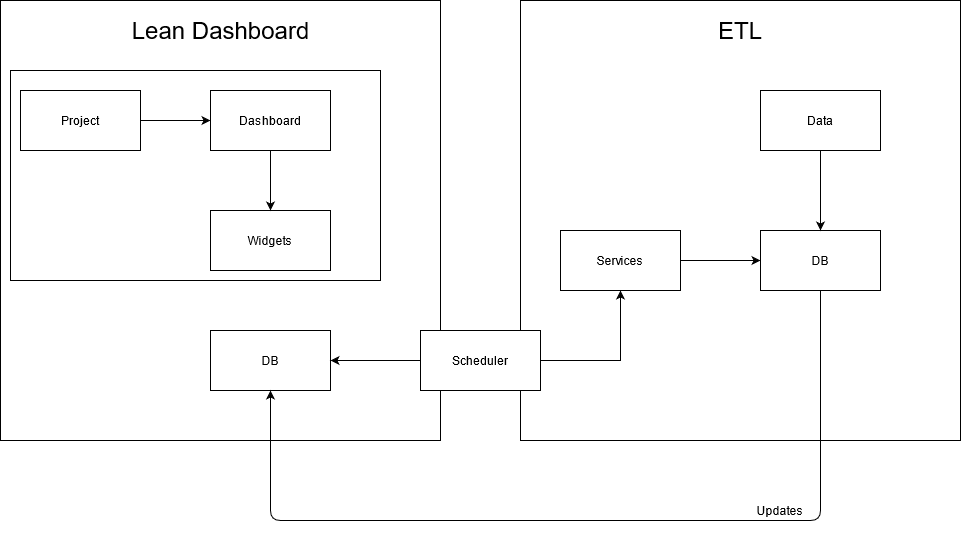
Um desenho inicial do fluxo de dados da aplicação Lean para o Scheduler.

Numa fase inicial, um dashboard não contem widget nenhum. O utilizador deve selecionar um widget dos que nós disponibilizamos para ser adicionado ao dashboard.

****

Este processo representa a adição de um widget ao dashboard de um projeto. Este widget é criado do lado do ETL, de modo a ultrapassar os problemas de tempo que seriam causados pelo scheduler. Assim que fosse feita a associação de um widget a um dashboard, este chamaria o ETL para a criação do mesmo em base de dados.

Após a adição de pelo menos um widget a um dashboard, o processo do scheduler começaria a funcionar.



Cada dashboard teria um processo scheduler associado, sendo que a informação de cada widget seria atualizada periodicamente. Este processo iria buscar informação sobre os widgets que o dashboard tem, executando as funções do ETL necessárias para que este seja atualizado.

**Exemplos de Widgets(Estado atual)**

**Widget Jira Sprint Gauge Chart**

"\_index": "etl-widgets",

                "\_type": "\_doc",

                "\_id": "E1-YhXkBDnV13TBBrohx",

                "\_score": 1.0,

                "\_source": {

                    "name": "Jira sprint gauge chart",

                    "code": "J\_SDGC",

                    "data": [

                        {

                            "sprintName": "SP Sprint 2",

                            "info": {

                                "remaining\_days": 14,

                                "past\_days": 14,

                                "difference\_in\_days": 28,

                                "percentage": 50

                            }

                        },

                        {

                            "sprintName": "PROJ Sprint 1",

                            "info": {

                                "remaining\_days": 14,

                                "past\_days": 14,

                                "difference\_in\_days": 28,

                                "percentage": 50

                            }

                        }

                    ]

Este widget teria informação suficiente para mostrar o número de dias que faltam até ao final do sprint num gauge chart.

**Widget Squash Test Results Pie Chart**

"\_index": "etl-widgets",

                "\_type": "\_doc",

                "\_id": "FF-bhXkBDnV13TBBc4hp",

                "\_score": 1.0,

                "\_source": {

                    "name": "Squash test results pie chart",

                    "code": "S\_PTPC-5",

                    "data": [

                        {

                            "total": 171,

                            "counts": [

                                {

                                    "status": "SUCCESS",

                                    "percentage": "18.13"

                                },

                                {

                                    "status": "READY",

                                    "percentage": "64.33"

                                },

                                {

                                    "status": "FAILURE",

                                    "percentage": "11.11"

                                },

                                {

                                    "status": "RUNNING",

                                    "percentage": "4.09"

                                },

                                {

                                    "status": "BLOCKED",

                                    "percentage": "2.34"

                                }

                            ]

                        }

                    ]

Este widget contém informação sobre os estados de todos os testes de um projeto Squash em percentagem para ser demonstrado num pie chart.

**Widget Squash Test Per Iteration Data Table**

"\_index": "etl-widgets",

                "\_type": "\_doc",

                "\_id": "FV-chXkBDnV13TBBkYgA",

                "\_score": 1.0,

                "\_source": {

                    "name": "Squash test per iteration data table",

                    "code": "S\_TIDT-5",

                    "data": [

                        {

                            "campaign": 18,

                            "iteration": "1",

                            "counts": [

                                {

                                    "status": "SUCCESS",

                                    "counts": 2

                                },

                                {

                                    "status": "READY",

                                    "counts": 2

                                }

                            ]

                        },

                        {

                            "campaign": 19,

                            "iteration": "Itération",

                            "counts": [

                                {

                                    "status": "SUCCESS",

                                    "counts": 2

                                },

                                {

                                    "status": "FAILURE",

                                    "counts": 2

                                },

                                {

                                    "status": "READY",

                                    "counts": 9

                                },

                                {

                                    "status": "RUNNING",

                                    "counts": 1

                                }

                            ]

                        },

Esta widget tem informação sobre cada iteração de um projeto, nomeadamente uma contagem dos vários estados dos testes de modo a os expor numa data table.

**ETL**

Alguns dos módulos do ETL foram melhorados de modo a melhor satisfazerem o seu propósito.

O módulo services agora é responsável por obter dados do módulo data e se necessário passar esses dados para o respetivo módulo transformer, executando depois as funções de armazenamento na base de dados do módulo DB.

O módulo DB foi alterado de modo a ser responsável exclusivamente pelo acesso à base de dados, sendo que qualquer método que envolva a transformação dos dados estará presente no módulo transformer, isto inclui a modificação de dados para ser transformados em widgets ou apenas a filtração das propriedades do objeto JSON obtido das várias fontes de informação.

**Scheduler**

De momento, já temos uma pequena implementação do Scheduler utilizando o módulo node Cron. Para cada dashboard é criado um job que é executado em intervalos de tempo definidos para teste. Dentro do módulo scheduler existe um mapa que associa um código criado por nós a cada função do módulo “services” do lado do ETL. Este tem como chave um código, uma string, e como valor a função do services a ser chamada.

O código definido é único para cada widget e contém informação relativa à fonte de informação, o widget que é necessário e os vários parâmetros necessários para uma chamada correta à função do ETL.

O job do scheduler executa uma função que itera pelos ids dos widgets associados a um dashboard, obtém a informação dos mesmos e através do código que estes têm na sua estrutura procura no mapa a função a ser chamada, executando-a com os parâmetros necessários e com o id do widget a atualizar informação.

Está também implementada uma função que altera a cadência da chamada à função do job.

Foram verificados os seguintes problemas:

* Os widgets estão já a ser adicionados. Verificámos uma dificuldade na associação de um widget criado e as informações necessárias para o scheduler criar um job para os widgets. Foi pensado no seguinte modelo para uma widget no futuro poder ser atualizada.

widget = {  
 id : 'id', name : 'Squash Test Pie Chart',  
 code : 'S\_ABCD-param1-param2-param3…',  
 projectInfo : {  
 azurePath : 'path',  
 jiraPath : 'path',  
 squashPath : 'path'  
 },  
 data : [. . .]  
}

* A cadência da atualização dos widgets deve ser global para um dashboard? Ou cada widget deve ter o seu próprio ritmo?
* Que tipo de informações precisaríamos de lhe passar para além dos caminhos para as API, tokens e cadência? Esta informação estaria presente no widget?

**Autenticação e Tokens**

Numa fase inicial, começámos apenas por suportar operações de criação de utilizadores, login e logout para utilizadores locais utilizando o módulo Authization. Apesar de termos tido inicialmente a ideia de armazenar nós próprios os utilizadores num índice elastic, percebemos que faria mais sentido utilizar a funcionalidade de utilizadores locais do Authizantion (guardando a sua informação numa base de dados Postgres).

Numa próxima fase queremos implementar os acessos/restrições utilizando o modelo RBAC do módulo, bem como definir roles e respetivas permissões. Isto iria fazer com que passássemos a verificar se o user autenticado tem acesso a um dado projeto, se é gerente desse mesmo projeto e o pode alterá-lo, se consegue ver os dashboards…

Surgiu a seguinte questão no que toca às credenciais da aplicação:

* A picture containing timeline

  Description automatically generatedOnde são guardadas as credenciais? Junto de um dado projeto ou de cada utilizador:

Guardar junto do projeto seria benéfico. Só seriam utilizados um par de credenciais (seriam sempre as do manager do projeto). As credenciais seriam inseridas pelo manager.

A picture containing chart

Description automatically generated

Guardar as credenciais junto de cada user (cada user teria as suas credencias) iria possivelmente causar mais informação a ser armazenada na db e a ser gerida pela aplicação.