**Apresentação de Progresso**

**Modelo de Dados**

Para realizar o armazenamento dos diversos dados optámos por utilizar o Elastic Search, tendo chegado ao seguinte modelo (ainda que sujeito a alterações):

Graphical user interface, application, Word

Description automatically generated

**Lean Projects**

- Está guardado no seu próprio index do elastic search, lean-projects

- Das propriedades apresentadas no esquema, Project Dashboards seria um array de Dashboards

**Dashboards**

- Não está guardado num índice. Um dashboard está apenas armazenado dentro de um projeto

- Das propriedades apresentadas no esquema, Widgets seria um array de ID’s das várias widgets presentes no dashboard

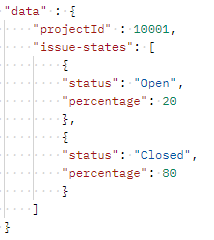
**Widgets**

Widgets são ainda um aspeto a discutir e perceber a sua organização. Encontram-se propositadamente de uma cor diferente para frisar que o seu desenho não está finalizado.

- Poderão estar no seu próprio index, etl-widgets. Sendo widgets o resultado do etl, poderão ficar no index do elastic etl-widgets e o index servir de ponto de ligação do etl e da aplicação. O Etl coloca os widgets, e estes por sua vez vão ser obtidos pela aplicação.

- Teriam um nome simples, para o utilizador poder com facilidade decidir os widgets pretendidos no seu dashboard, um id para poderem ser acedidos com facilidade e um código para identificar os widgets e os parâmetros necessários para o ETL os atualizar. Data seria um array ou um objeto que contivesse a informação necessária para representar a informação necessária, por exemplo, no caso da representação de um pie chart este teria o seguinte aspeto:

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



**Hierarquia de dados**

Graphical user interface, application

Description automatically generated

Das várias fontes que o coordenador nos propôs nós selecionámos 3 que consideramos ser importantes. Jira, Squash e Azure.

A plataforma Jira serve para coordenar as equipas e contêm informação sobre issues, sprints etc.

A plataforma Squash serve principalmente para organizar os resultados dos vários testes que os softwares a desenvolver efetuem.

A plataforma Azure é semelhante à do Jira, oferecendo também suporte para os seus próprios testes.

**Potênciais Widgets:**

**Jira**

**Widget Completed Issues**

Atributos necessários: (key, summary, issueType, Priority, Status, Story Points)

Tipo de Widget: Data Table Widget

**Widget Status Issues**

Atributos necessários: (Issue state, created, ended)

Tipo de Widget: Date Bar Chart Widget

data: Week, Issue count (based on the issue status)

**Widget Sprint**

Atributos necessários: (Id, name, state, startDate, endDate, goal)

Tipo de Widget: Data Table Widget

**Widget Items at Risk**

Necessita do plugin. Se com o plugin for possível obter o risco de cada issue, é relativamente fácil:

Atributos necessários: (isAtRisk, count)

Tipo de Widget: Gauge chart Widget

**Widget Burndown Chart**

Nos links testados não está a ser dada a informação que achamos necessária. Era de esperar ter objetos para um gráfico (como coordenadas x e y, e o que representam) mas são mostrados detalhes como issues e sprints. Api também não menciona como obter informação do Burndown Chart

**Widget Velocity Chart**

Atributos necessários: (sprint name, sprint id, issue id, issue state)

Tipo de Widget: Date Bar Chart Widget

data: Sprint, total issues, completed issues

**Widget Cumulative Flow Diagram**

(Não conseguimos ver um exemplo)

Atributos necessários: (issue state, date)

Tipo de Widget: Graph

data(months, issue state, count)

**Widget High Level Status**

Não conseguimos ver um exemplo, nem perceber a informação necessária a ser obtida para a sua representação

**Widget Remaining Estimate**

Não conseguimos ver um exemplo, nem perceber a informação necessária a ser obtida para a sua representação

Total Widget Jira: 9

**Squash**

**Widget Status Test**

Atributos necessários: (test id, test name, status)

Tipo de Widget: Pie Chart

data: status, percentage

**Widget Test per Iteration**

Atributos necessários: (iteration\_name, executing, on going, success, arbitrator, failure, block, not tested, Number of planned tests,Progress of execution)

Tipo de Widget: Data Table Chart

Squash: 2(com a data table conseguimos mostrar bastante informação)

**Azure**

**Widget Test Case Status**

Atributos necessários: (test id, test name, status)

Tipo de Widget: Data Bar Graph

data: status, percentage

**Widget Sprint Goal + Countdown**

Atributos necessários (sprint id, sprint name, sprint description, days remaining to end)

Tipo de Widget: Data Table Widget

**Widget Active Tasks**

Atributos necessários (task id, task name, date, status)

Tipo de Widget: : Data Table Widget

Azure: 3

**Exemplos de Widgets(Já implementados)**

Faltam alguns detalhes como o código

**Widget Jira Sprint Gauge Chart**

"\_index": "etl-widgets",

                "\_type": "\_doc",

                "\_id": "hgoPYHkBcUCkGPbawwET",

                "\_score": 1.0,

                "\_source": {

                    "name": "Jira sprint gauge chart",

                    "data": [

                        {

                            "sprintName": "SP Sprint 2",

                            "info": {

                                "remaining\_days": 21,

                                "past\_days": 7,

                                "difference\_in\_days": 28,

                                "percentage": 25

                            }

                        },

                        {

                            "sprintName": "PROJ Sprint 1",

                            "info": {

                                "remaining\_days": 21,

                                "past\_days": 7,

                                "difference\_in\_days": 28,

                                "percentage": 25

                            }

                        }

                    ]

                }

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": "hwoTYHkBcUCkGPba5gFs",

                "\_score": 1.0,

                "\_source": {

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

                    "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": "iAoWYHkBcUCkGPbaBgHu",

                "\_score": 1.0,

                "\_source": {

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

                    "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

                                }

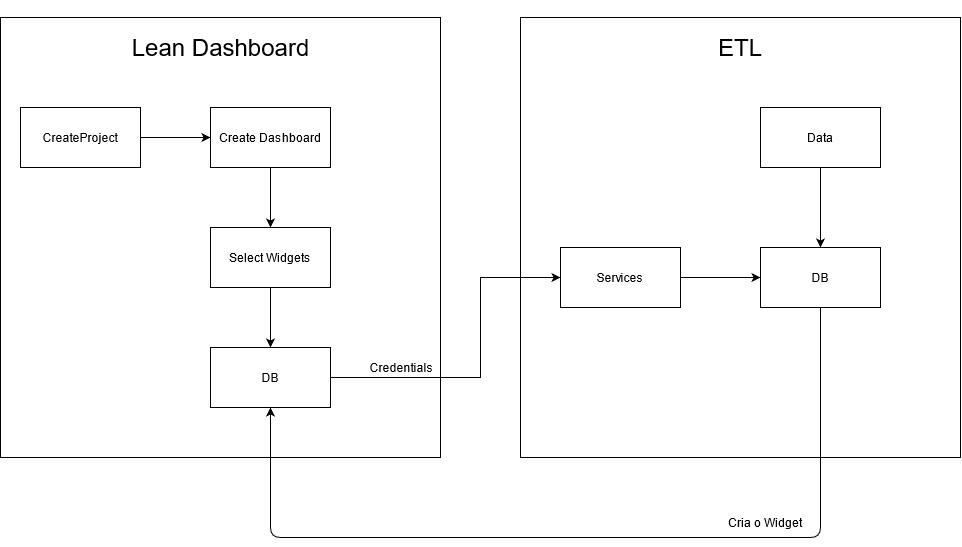
                            ]

                        } ]

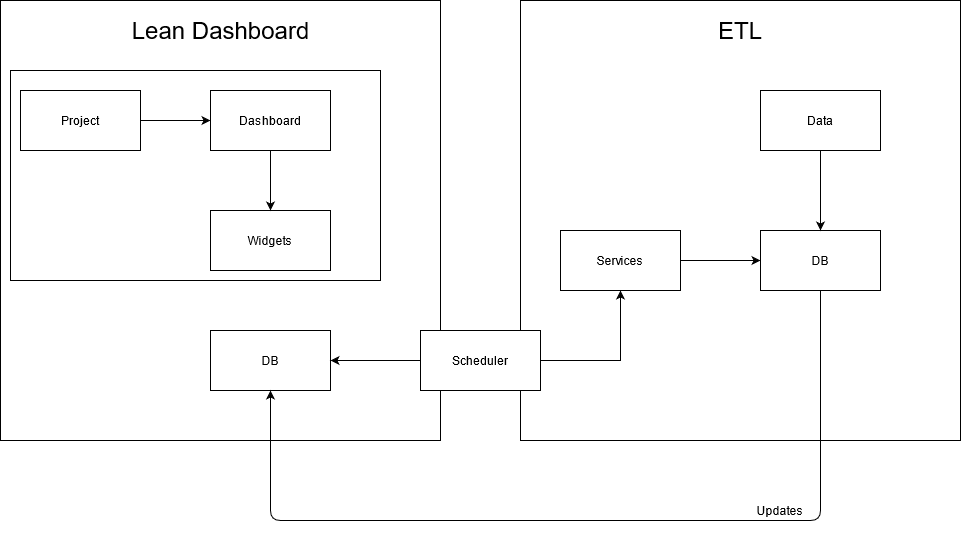
Esta widget tem informação sobre cada iteração de um projeto, nomeadamente uma contagem dos vários estados dos testes de modo a os expor numa data table.

**Fluxo de dados/Estrutura do código**

Um desenho inicial do fluxo de dados da aplicação Lean.

****

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.

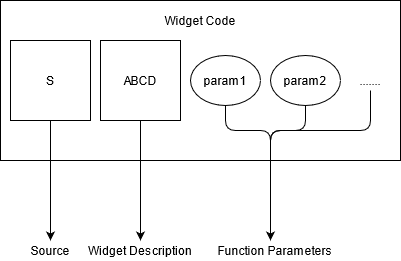


Após a criação de pelo menos um widget do lado do ETL, o processo do scheduler começaria a funcionar. Cada dashboard teria um processo associado, sendo que a informação de cada widget seria atualizada periodicamente. Este processo iria buscar informação sobre os widgets que o dashboard tem, executando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

**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 deste scheduler existe ainda um mapa que associa um código criado por nós a cada função do módulo “services” do lado do ETL.

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



**Autenticação e Tokens**

Numa fase inicial, começámos apenas por suportar operações de criação de utilizadores, login e logout para utilizadores locais. Num futuro próximo, e depois de esclarecidas algumas questões sobre tokens e credenciais, começaremos a implementação do módulo Authization.

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.