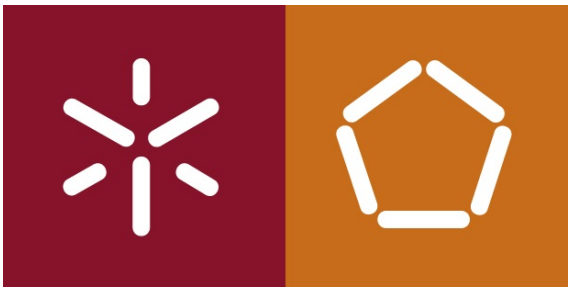

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Aplicações Informáticas na Biomedicina

A realização de urgências gerais num determinado hospital nacional

Adriana Meireles (A82582)
Bárbara Cardoso (A80453)
Carla Cruz (A80564)
Inês Alves (A81368)
Shahzod Yusupov (A82617)

17 de Dezembro de 2019

Resumo

No âmbito da Unidade Curricular de Aplicações Informáticas na Biomedicina, complementar do Mestrado em Engenharia Informática, foi-nos proposto o desenvolvimento de um trabalho prático tendo como base o tema de trabalho: A realização de urgências gerais num determinado hospital nacional.

Conteúdo

1 Introdução

Nesta Unidade Curricular foi-nos proposta a realização de um trabalho prático com a finalidade de consolidar os conhecimentos adquiridos sobre as diversas ferramentas abordadas nas aulas.

Para tal foi nos disponibilizado pelos docentes um ficheiro CSV, nomeadamente urg_inform_-geral.csv, que contém dados reais da lista de realização de urgências gerais num determinado hospital nacional:

- URG_EPISODIO - Número de episódio da urgência (identificador único)
- DATAHORA_ADM - Data e hora da admissão do paciente
- DATAHORA_ALTA - Data e hora da alta do paciente
- ALTA_DES_ESPECIALIDADE - Descrição da especialidade
- DES_LOCAL - Descrição do local
- DES_PROVENIENCIA - Descrição da proveniência
- SEXO - Género
- DTA_NASCIMENTO - Data de nascimento
- DES_CAUSA - Descrição da causa

Numa primeira fase, estes dados,juntamente com o modelo dimensional criado, são usados para o povoamento de um data warehouse. Este povoamento é feito usando dois procedimentos distintos: com jobs em Talend e com uma script em SQL.

Por fim, recorrendo à ferramenta Power BI e ao data warehouse, são criados alguns indicadores clínicos que poderão vir a ser úteis para uma aplicação informática direcionada às urgências gerais no hospital em questão.

2 Data Warehouse

Realizada uma análise dos campos presentes e dos dados apresentados pelos docentes, foi possível a criação do seguinte *Data Warehouse*:

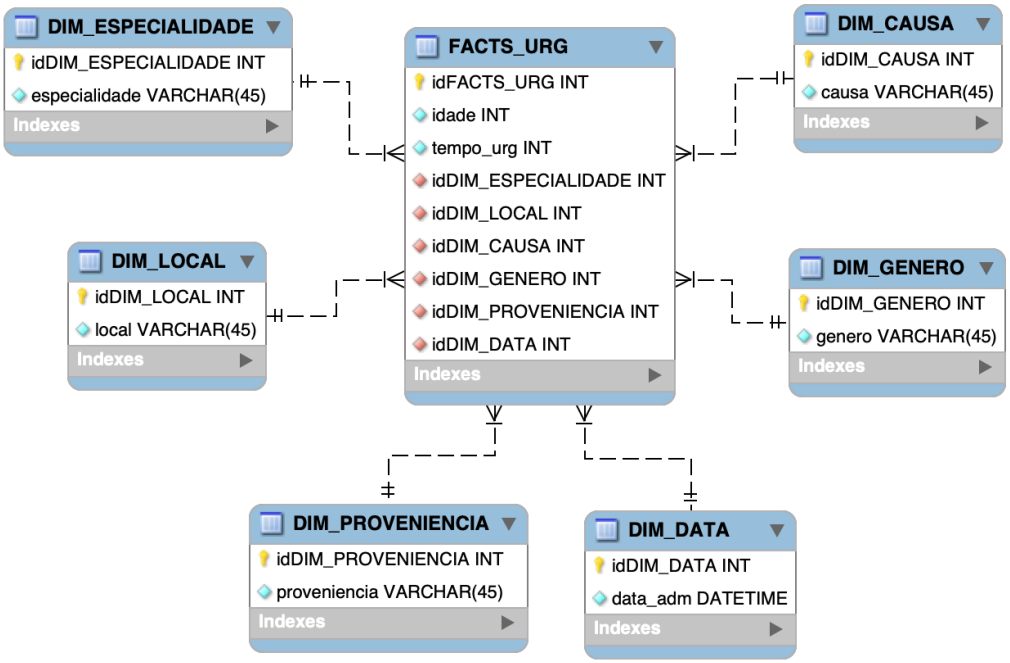

Figura 1: *Data Warehouse* - modelo em estrela

Na visão geral apresentada identificamos campos que não foram incluídos, nomeadamente a **data de alta** e a **data de nascimento** ficando excluídas da tabela de factos e da tabela dimensão relativa à data (DIM_DATA). Estas não pertencem à dimensão de datas, dado que apenas deve constar a data de entrada no registo do serviço referente ao facto relacionado, pelo que a data de admissão é a que nos importa para o caso de estudo.

Ainda assim, estas informações (data de nascimento do utente e data de alta) não são completamente perdidas, dado que podemos sabê-las a partir da idade (concluindo qual a sua data de nascimento) e do tempo de admissão (obtendo a data de alta do paciente).

A restante informação encontra-se representada nas dimensões existentes e aqui ilustradas.

É ainda importante referir que foi necessário tornar todos os nossos *IDs* auto incrementáveis (através da opção *autoincrement* do *MySQL Workbench*), uma vez que, ao popular as tabelas, se tal opção não estivesse selecionada, encontraríamos um erro, já que não existiria um valor *default* para este campo.

3 Povoamento

Tal como pedido no enunciado deste projeto, foram utilizadas duas técnicas de povoamento do *Data Warehouse* mencionado:

- **Em SQL com uma *script***;
- **Com *jobs* na ferramenta *Talend***

3.1 SQL

Há diversas formas de realizar o povoamento das tabelas referidas utilizando o *MySQL Workbench*, no entanto, a forma mais fácil de realizar este processo foi através da utilização de cursores.

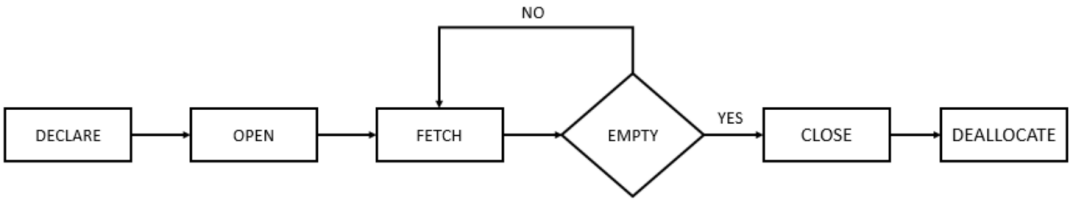

Figura 2: Ciclo de vida de um cursor

Um cursor é uma instrução *SELECT* que será acedida linha a linha através de um ciclo *while* e alguns comandos específicos para cursores. Estes são, normalmente, utilizados em *stored procedures*. Isto significa que o código aqui escrito pode ser guardado e reutilizado as vezes que forem necessárias.

Assim, foram criados cursores para povoar as tabelas de dimensão e um outro para povoar a tabela de factos. A título de exemplo para um cursor cujo objetivo era povoar uma tabela de dimensão, vejamos o seguinte código presente no *script insert_causa()*:

```
CREATE DEFINER=‘root‘@‘localhost‘ PROCEDURE ‘insert_causa‘()
BEGIN
    declare done boolean default false;
    declare causa varchar(45);
    declare cur1 cursor for select distinct DES_CAUSA from DB_URG.urg_inform_geral;
    declare continue handler for not found set done = true;

    open cur1;

    read_loop: loop
        fetch cur1 into causa;

        if done then
            leave read_loop;
        end if;

        insert into DW_URG.DIM_CAUSA (causa) values (causa);

    end loop;
```

```
    close cur1;
END
```

Como podemos imaginar, todas as outras tabelas de dimensão têm um raciocínio análogo, pelo que não se justifica fazer aqui a demonstração do código dos restantes cursores. No entanto, para povoar a tabela de factos, temos um cursor já com algumas diferenças, uma vez que necessitamos de inserir dados com as *foreign keys* definidas previamente. Para além disso, também precisavámos de processar informações de modo a obter a idade de um paciente, bem como o tempo que este residiu na urgência do hospital.

```
CREATE DEFINER=‘root‘@‘localhost‘ PROCEDURE ‘insert_facts‘()
BEGIN
    declare done boolean default false;
    declare id int;
    declare idade int;
    declare tempo_urg int;
    declare id_CAUSA int;
    declare id_DATA int;
    declare id_ESPECIALIDADE int;
    declare id_GENERO int;
    declare id_LOCAL int;
    declare id_PROVENIENCIA int;

    declare cur1 cursor for SELECT GERAL.URG_EPISODIO, TIMESTAMPDIFF(YEAR,
    GERAL.DTA_NASCIMENTO, GERAL.DATAHORA_ALTA), TIMESTAMPDIFF(MINUTE, GERAL.DATAHORA_ADM,
    GERAL.DATAHORA_ALTA), DC.idDIM_causa, DD.idDIM_data, DE.idDIM_especialidade,
    DG.idDIM_genero, DL.idDIM_local, DP.idDIM_proveniencia

    FROM dw_urg.dim_causa DC, dw_urg.dim_data DD, dw_urg.dim_especialidade DE,
    dw_urg.dim_genero DG, dw_urg.dim_local DL, dw_urg.dim_proveniencia DP,
    DB_URG.urg_inform_geral GERAL

    WHERE DC.causa = GERAL.DES_CAUSA
    AND DD.data_adm = GERAL.DATAHORA_ADM
    AND DE.especialidade = GERAL.ALTA_DES_ESPECIALIDADE
    AND DG.genero = GERAL.SEXO
    AND DL.local = GERAL.DES_LOCAL
    AND DP.proveniencia = GERAL.DES_PROVENIENCIA ;

    declare continue handler for not found set done = true;

    open cur1;

    read_loop: loop
    fetch cur1 into id, idade, tempo_urg, id_CAUSA, id_DATA, id_ESPECIALIDADE,
                    id_GENERO, id_LOCAL, id_PROVENIENCIA;

        if done then
            leave read_loop;
        end if;

        insert into dw_urg.facts_urg(idFACTS_URG, idade, tempo_urg, idDIM_ESPECIALIDADE,
                                     idDIM_LOCAL, idDIM_GENERO, idDIM_CAUSA,
```

```
                                        idDIM_PROVENIENCIA, idDIM_DATA)
                values (id, idade, tempo_urg, id_ESPECIALIDADE, id_LOCAL, id_GENERO,
                        id_CAUSA, id_PROVENIENCIA, id_DATA);

    end loop;

    close cur1;
END
```

3.2 Talend

Passamos agora para o segundo método de povoamento: *jobs* fazendo uso da ferramenta *Talend*. Tal como na técnica anterior, também aqui podemos dividir o povoamento em duas partes:

1. **povoamento das tabelas de dimensão**;
2. **povoamento da tabela de factos**

Ora, começando pelo povoamento das tabelas de dimensão podemos observar o seguinte *job* criado para o efeito:

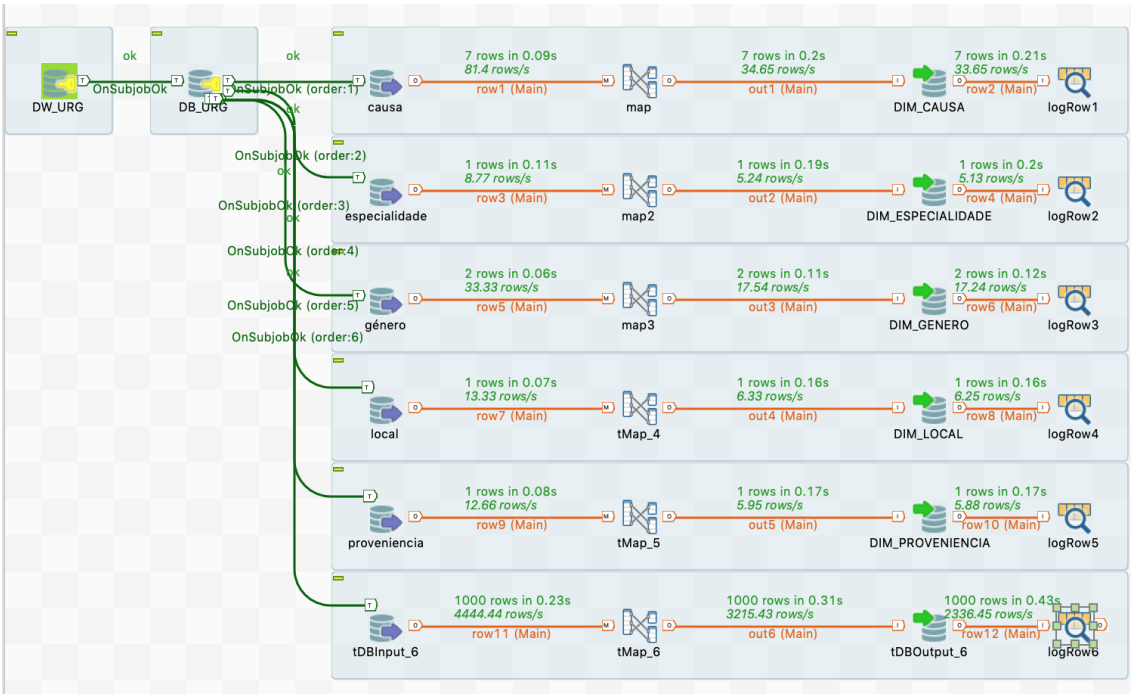

Figura 3: *Job* para povoamento das tabelas de dimensão

Este *job* realiza o povoamento das tabelas de dimensão fazendo uso de *queries* SQL que seguem o seguinte formato:

```
SELECT DISTINCT 'campo' FROM DB_URG.urg_inform_geral
```

Como podemos inferir, esta *query* permite-nos selecionar sem repetições o **campo** que queremos. Posteriormente, o resultado desta seleção é enviado para um componente *tMap* que irá realizar o mapeamento necessário. Por fim, o resultado é enviado para a tabela pretendida.

Apesar de possuirmos *IDs* auto incrementáveis, deparamo-nos com alguns problemas aquando da sua visualização, uma vez que alguns se mantinham sempre a 0. Para corrigir este problema e garantir que os *IDs* estavam corretos optamos por usar a seguinte expressão em todos os campos que faziam referência a um *ID*:

```
Numeric.Sequence('id_campo', 1, 1)
```

Como podemos observar pela imagem acima, fazemos ainda uso de componentes *tLogRow* a fim de tornar a visualização dos dados mais percetível.

```
.--------------------+--------.
|            logRow1           |
|=--------------------+-------=|
|causa                |id_causa|
|=--------------------+-------=|
|DOENCA               |1       |
|ACIDENTE DE TRABALHO|2       |
|OUTRAS               |3       |
|ACIDENTE ESCOLAR     |4       |
|ACIDENTE PESSOAL     |5       |
|ACIDENTE DE VIACAO   |6       |
|QUEDA                |7       |
'--------------------+--------'
```

Figura 4: Exemplo de LogRow para visualização de dados

Estando todas as tabelas de dimensão povoadas, passamos agora para a inserção de dados na tabela de factos.

Este *job* possui um raciocínio bastante parecido ao anterior, excetuando alguns pequenos pormenores nos quais iremos atentar.

Utilizando, novamente, um *tMap* podemos realizar a procura na tabela de dimensão pretendida. Para além disso, é também aqui que podemos visualizar o cálculo da idade de um utente:

```
Math.round(TalendDate.diffDate(TalendDate.getCurrentDate(),
          row1.DTA_NASCIMENTO,"dd") / 365 )
```

E do tempo que este esteve na urgência do hospital:

```
TalendDate.diffDate(row1.DATAHORA_ALTA,row1.DATAHORA_ADM,"mm")
```

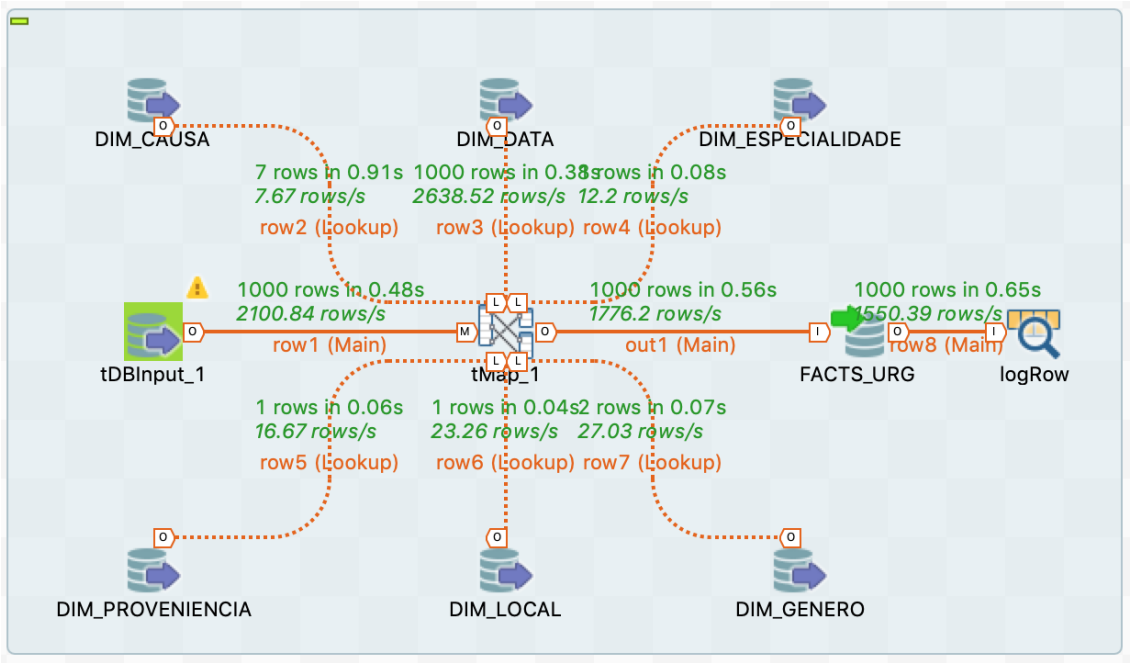

Figura 5: *Job* para povoamento da tabela de factos

3.3 Vantagens e desvantagens dos processos de povoamento referidos

É notória a diferença entre ambas as técnicas utilizadas, já que utilizam ferramentas e técnicas de *coding* diferentes.

No povoamento feito em SQL, onde optamos por utilizar cursores, podemos perceber que o povoamento das tabelas de dimensão é bastante simples e monótono, isto é, fazendo para uma tabela, todo o processo se mantém para as restantes. No entanto, o povoamento da tabela de factos revelou-se bastante mais complicado, sendo necessária uma atenção extra aos seus detalhes. Assim, podemos perceber que povoar o *Data Warehouse* utilizando SQL é uma técnica bastante ineficiente, já que requer um tempo dispendido bastante elevado e até diversas tentativas para a sua correta implementação.

Já no caso da ferramenta *Talend* tudo se torna mais simplificado. De um ponto de vista geral, com alguma experiência no manuseamento da ferramenta (adquirida nas aulas práticas da unidade curricular mencionada) o processo de povoamento é bastante rápido. Com este método não há a necessidade de nos preocuparmos com código, sendo apenas necessárias algumas expressões matemáticas já referidas nas secções anteriores, mas que se encontram facilmente, graças à boa documentação da ferramenta. Para além disso, também o seu design graficamente apelativo representa uma vantagem na utilização desta ferramenta. Assim, é seguro afirmarmos que a utilização do *Talend* em detrimento do *SQL* é uma boa opção, poupando o programador em tempo e cansaço psicológico, já que todas as operações necessárias se encontram aqui simplificadas.

4 Indicadores Power BI

O *Power BI* foi uma ferramenta utilizada, como já fora previamente. Esta fornece visualizações interativas com uma interface simples, permitindo também partilhar informação com outros utlizadores ou até incorporá-la num *Website*, bem como numa aplicação móvel. Permite, assim, a ligação a diferentes tipos de fontes de informação e, consequentemente, gerar novo conhecimento através de relatórios e *dashboards*.

Para tal criamos alguns indicadores de interesse clínico. Começamos por escolher indicadores relacionados ao género tais como o número de pessoas do género feminino e masculino que frequentaram as urgências e, de seguida, com base nas causas que levaram as pessoas ao hospital, o número de pessoas que entraram tendo esses motivos, bem como a distinção entre ambos os géneros. Esta análise foi importante pois permite analisar as principais causas que levam as pessoas a deslocarem-se às urgências, assim como o género mais afetado. Apresentamos de seguida os indicadores do *Power BI* realizados com base no explicado.

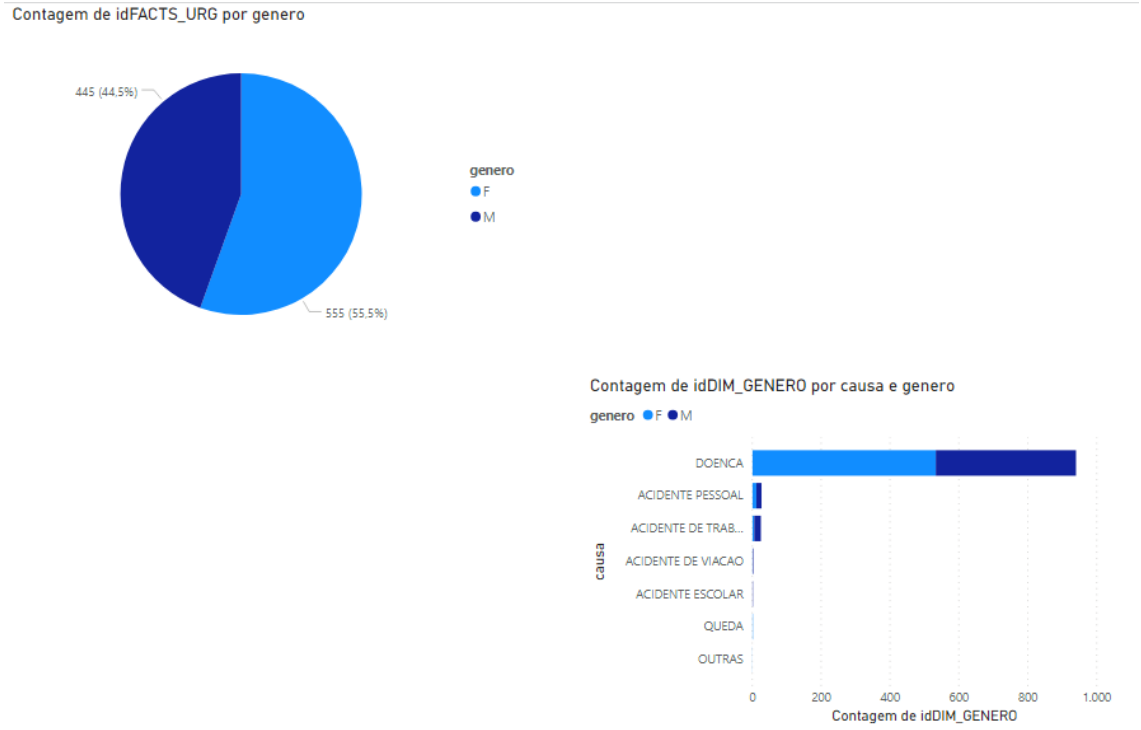

Figura 6: Indicadores Clínicos relacionados com o género

De seguida, achamos que seria interessante focar os indicadores clínicos nas médias como, por exemplo, um paciente ter uma causa que o leva às urgências e, com base na mesma, qual a média de tempo de espera. Achamos que também seria benéfico uma análise da média de idades por causa, isto é, qual a idade média que costuma apresentar determinada causa com mais frequência. Estes indicadores clínicos podem ser vantajosos na medida em que possibilitam o melhoramento das condições de espera com base nas idades bem como, por exemplo, colocar mais pessoas especializadas a receber certo tipo de problema para a espera não ser tão elevada.

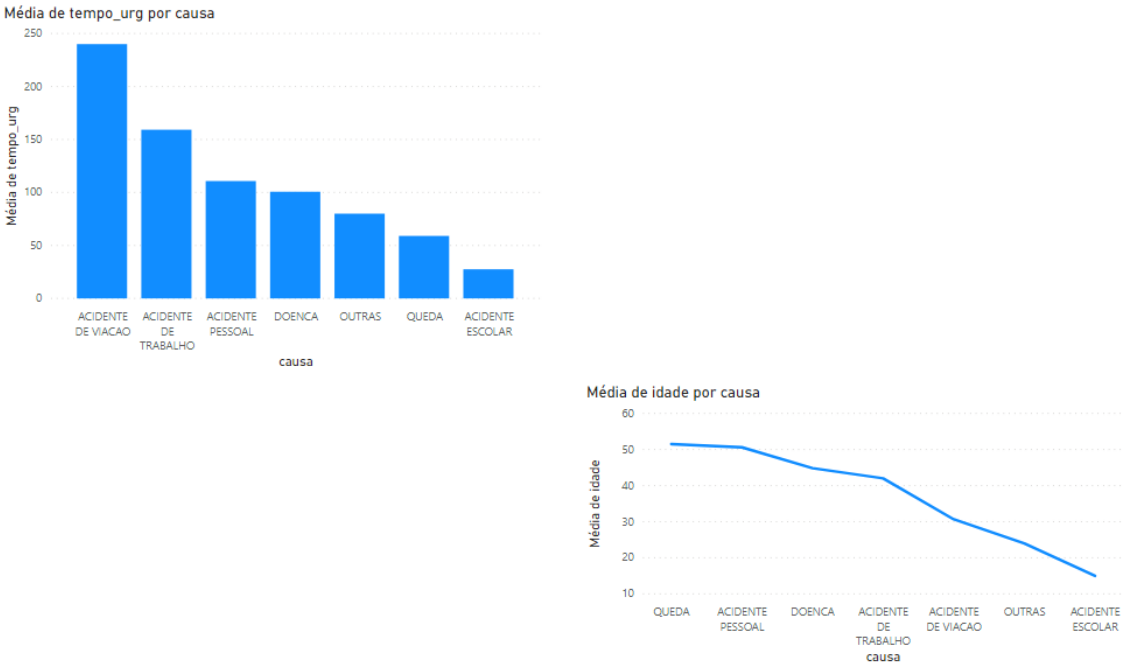

Figura 7: Indicadores Clínicos relacionados com as médias

Por fim, achamos bastante relevante este indicador clínico. O mês analisado neste trabalho prático é o de janeiro, fazendo assim uma análise mais pormenorizada desse mês. Desta forma, é visualizada, por dia, o número de entradas nas urgências (a soma das causas por dia) e diferenciando, em cada dia, as diferentes causas que fez com que estes utentes dessem entrada no hospital. Desta forma, é possível ver as diferentes causas nos diferentes dias que levaram os doentes às urgências e promover uma melhor gestão da procura, de forma a dar uma melhor resposta aos mesmos.

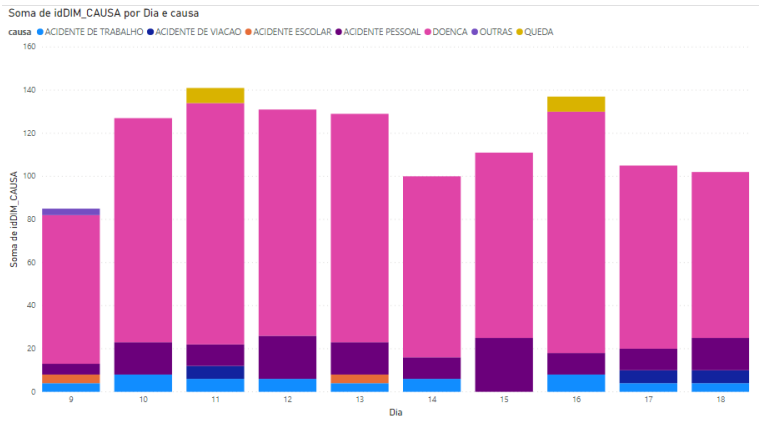

Figura 8: Indicadores Clínicos relacionados com as causas

De seguida encontra-se a *dashboard*:

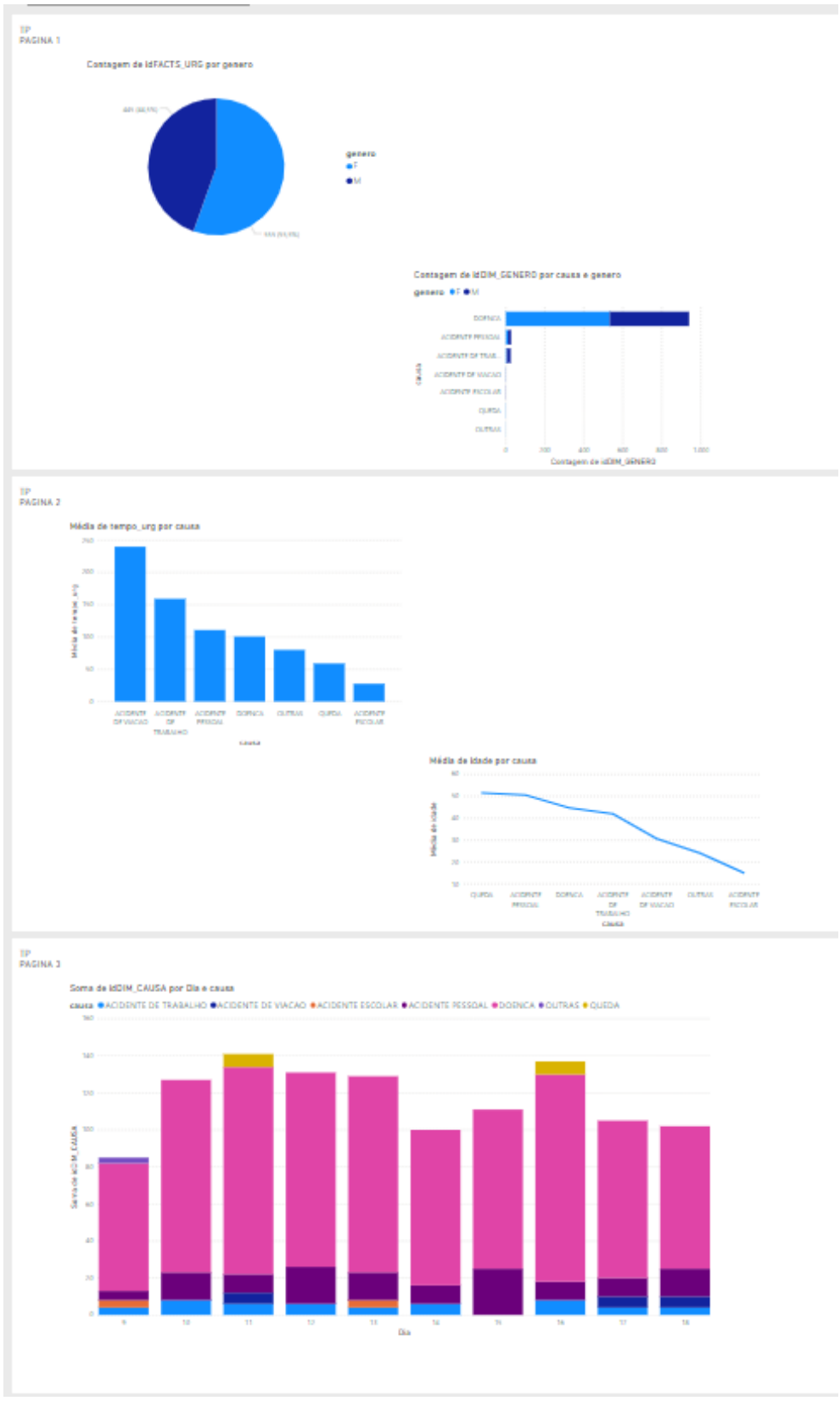

Figura 9: Dashboard

5 Indicadores Clínicos

Iremos agora exibir alguns exemplos de como poderiam ser utilizados os indicadores clínicos criados de forma útil. Vamos mostrar como poderiam estes ser apresentados aos utilizadores de um programa de gestão do serviço de urgências num hospital. O principal objetivo desta aplicação é ser intuitiva e de fácil manuseamento em que o principal foco é a análise estatística dos dados sobre as urgências dum hospital nacional.

Esta aplicação terá variadas funcionalidades que poderão ser usadas tanto num computador como num telemóvel.

Para além dos dados servirem para estastísticas, estes irão servir também de grande importância para o processo de triagem, em que há uma separação imediata entre os pacientes conforme a sua urgência. Assim, evita-se, por exemplo, que pessoas que estejam com gripe ou algum tipo de doença contagiosa peguem a outras pessoas com algum tipo de trauma. Outro exemplo em que esta aplicação poderá ser de grande utilidade é no fornecimento de informações aos pacientes como, por exmeplo, o tempo de espera para a urgência em causa. Isto é importante, pois caso o hospital esteja com um elevado número de pacientes, estes poderão optar por esperar ou ir para outro hospital. Também a informação relativa à idade média correspondente será útil para o médico analisar os dados e poder dar informação a um paciente idoso, por exemplo, sobre o tipo de doenças a que ele pode estar sujeito.

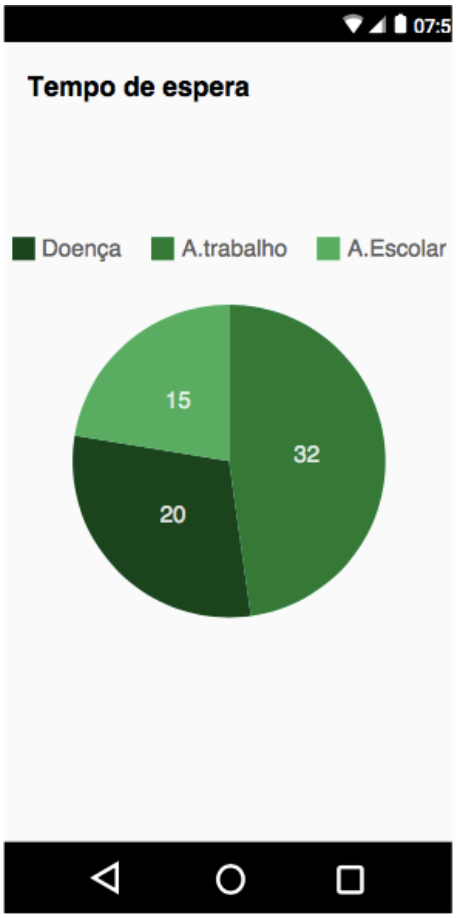

Figura 10: Exemplo de *mockup* do sistema I

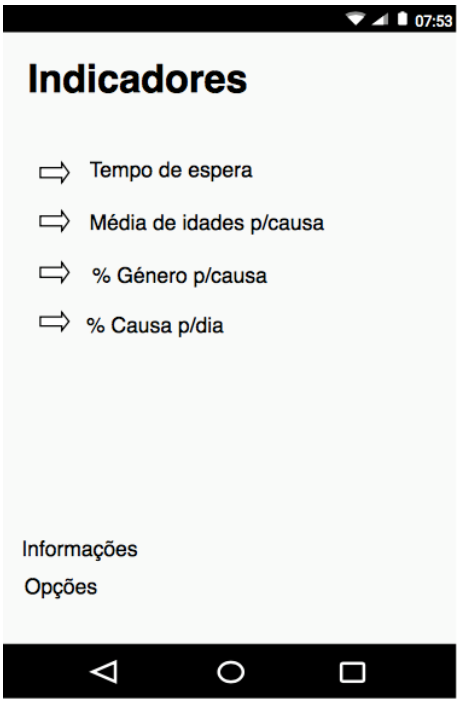

Figura 11: Exemplo de *mockup* do sistema II

6 Conclusão

Com a elaboração deste projeto, tornou-se evidente a importância que as ferramentas abordadas têm para a implementação de uma aplicação bem desenvolvida num data warehouse. Com especial atenção, as ferramentas Talend Open Studio e MySQL, em que podemos observar como programas totalmente diferentes extraíram, transformaram e carregaram os dados para o utilizador poder analisar e desenvolver de uma forma bastante eficaz.

No geral, ao longo da realização do trabalho, o grupo apenas sentiu algumas dificuldades no uso da ferramenta Talend, sendo esta ferramenta uma novidade.

Assim, reconhecemos como concluída esta etapa da unidade curricular em que foram aplicados todos os conhecimentos adquiridos ao longo do semestre neste projeto.