Mestrado em Engenharia Informática
Estágio
Relatório Final

Projeto DW-UC

Desenvolvimento de uma Data Warehouse para a Universidade de Coimbra
Área C

Hugo Figeiredo Costa
hfcosta@student.dei.uc.pt

Orientador:
Prof. Dr. Bruno Cabral
Data: 01 de Julho de 2014

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Agradecimentos

Ao meu orientador, Professor Doutor Bruno Cabral, pela ajuda, disponibilidade, paciência e acompanhamento durante todo o estágio.

Ao Engenheiro Pedro Pinto, pelo acompanhamanto prestado e pela disponibilidade.

À equipa de trabalho do projeto pela amizade e disponibilidade para que o projeto se concluísse com sucesso.

À equipa do Nónio pela ajuda prestada.

Aos meus pais pela compreensão e pelo apoio dado.

Resumo

Nos dias de hoje, há, cada vez mais, a necessidade de tomar decisões fundamentadas sobre a gestão dos recursos económicos, financeiros, humanos e patrimoniais presentes numa instituição.

No plano estratégico e de ação da Universidade de Coimbra, encontra-se definido um conjunto de indicadores, relativos a receita, que são necessários para apoiar as decisões tomadas pelo Conselho de Gestão. No entanto, para se obterem os valores correspondentes aos indicadores, é necessário que estes sejam calculados a partir dos vários sistemas fonte disponíveis na Universidade. Esta tarefa é feita manualmente e, por conseguinte, torna-se mais morosa e menos fiável.

Com o objetivo de colmatar o problema exposto, o presente estágio visa desenvolver uma *data mart,* onde seja possível consultar indicadores referentes à receita de propinas, de emolumentos, de juros e à utilização dos meios de pagamento. Desta forma, foram desenvolvidos vários *dashboards* que possibilitam aos utilizadores uma visualização interativa e atrativa dos vários indicadores. Os *dashboards* encontram-se disponíveis sob a forma de páginas *web*, o que permite que os resultados da análise estejam sempre disponíveis, independentemente da hora e do lugar em que são visualizados.

Palavras-Chave

Data mart, *Key Performance Indicators*, Universidade de Coimbra, *Business Intelligence*, Receita de Propinas, Receita de Emolumentos, *Dashboards*, *OLAP*, *ETL*

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

Abreviatura	Descrição
ATM	*Automated Teller Machine*
BI	*Business Intelligence*
CSV	*Comma Separated Value*
ER	*Entity Relationship*
ERP	*Enterprise Resource Planner*
ETL	*Extraction Transformation Loading*
OLAP	*Online Analytical Processing*
GSIIC	*Gestão de Sistemas e Infra-Estruturas de Informação e Comunicação*
MDX	*Multidimensional Expressions*
Non-RDBMS	*Non-Relational Database Management System*
PDF	*Portable Document Format*
PNG	*Portable Network Graphics*
RDBMS	*Relational Database Management System*
SaaS	*Software as a Service*
SQL	*Structured Query Language*
UC	*Universidade de Coimbra*
XML	*Extensible Markup Language*

Capítulo 1
Introdução

Com base em informação disponível, existe, atualmente, uma necessidade premente de um organização tomar decisões de gestão. A Universidade de Coimbra não é uma instituição diferente das outras e, por isso, a equipa reitoral decidiu aprovar o projeto SAMA, que visa a consolidação e a melhoria dos sistemas de apoio à modernização administrativa.

Dentro deste projeto, está, entre outras áreas de análise, a definição de indicadores para a monitorização da qualidade pedagógica e do desempenho dos serviços da Universidade de Coimbra. O projeto DW-UC pretende responder a esta necessidade, através do desenvolvimento de uma *data warehouse* para quatro áreas distintas.

Um dos objetivos primordiais de um projeto de *data warehouse* é poder oferecer aos seus utilizadores acesso a uma visão homogénea e centralizada de todos os processos de negócio primários da instituição, isto é, de todos os processos de negócio que geram receita para a instituição. O segundo objetivo mais importante, que um sistema deste tipo pode oferecer, é haver num só sistema uma consolidação de dados de outros sistemas, fornecendo, assim, uma única visão de todos os processos de negócio na *data warehouse* modelados. Trata-se de um projeto que é constituído por quatro grandes módulos, que contemplam a extração dos dados dos sistemas fonte, o seu armazenamento num sistema durável, a construção de um cubo *OLAP* e a sua apresentação num relatório ou em *dashboards*.

A *data warehouse* para a Universidade de Coimbra a ser desenvolvida, no âmbito do projeto UC-num, contempla já um conjunto de áreas que foram identificadas pela equipa reitoral como sendo prioritárias. Uma dessas áreas, a apresentada neste estágio, engloba a receita de propinas e emolumentos. As restantes áreas identificadas são a do sucesso escolar, a dos recursos humanos e a dos custos com o ensino, que se encontram em desenvolvimento por mais três colegas, também no âmbito de estágio. A *data warehouse* não é mais que um sistema capaz de armazenar grandes quantidades de dados, relativas a todas as atividades de uma organização. Trata-se de um sistema, onde, se for necessário, os dados podem ser total ou parcialmente partilhados entre as várias atividades de negócio.

Faz também parte, do âmbito deste estágio, o desenvolvimento de uma *data mart* para guardar os dados essenciais para calcular um conjunto de indicadores, relacionados com a receita de propinas e os respetivos juros e emolumentos, pagos pelos estudantes ou por entidades terceiras. Faz, ainda, parte, deste módulo de estágio, o cálculo de indicadores referentes aos meios de pagamento e postos de recebimento, utilizados pelas entidades pagadoras. A *data mart* não é mais do que um subconjunto da *data warehouse*, o que, neste caso, se traduz no desenvolvimento das estruturas necessárias para suportar os processos de negócio descritos.

A equipa de projeto é constituída pelo Professor Doutor Bruno Cabral, no papel de orientador, pelo Engenheiro Pedro Pinto, na qualidade de orientador do módulo do sucesso escolar, e por três colegas responsáveis por cada um dos módulos acima referidos. Dado que é do sistema Nónio que serão obtidos todos os dados necessários, o local de trabalho deste estágio situa-se no mesmo espaço onde se encontra a equipa de desenvolvimento do sistema Nónio.

1.1. Objetivos

Um dos objetivos do presente estágio passa por implementar uma *data mart*, relativa à receita de propinas e aos juros que lhe estão associados, à receita de emolumentos e às taxas de utilização dos meios de pagamento, disponibilizados pela Universidade de Coimbra. Para isso, será definido um conjunto relevante de indicadores que permitam mensurar o estado da cobrança de propinas, dos emolumentos, dos juros e da utilização dos meios de pagamento. Estes indicadores devem ser, posteriormente, apresentados em *dashboards*, onde possam ser acedidos e a sua evolução possa ser vista em tempo quase real.

Para que se seja possível definir os indicadores para esta *data mart*, é necessário, antes de mais, especificar um conjunto de questões às quais este módulo deverá ser capaz de responder de forma eficaz e assertiva. De seguida, apresenta-se uma amostra do tipo de questões que podem ser definidas. No anexo 1, encontra-se uma lista dos indicadores considerados para este projeto.

- Qual o número de entidades em incumprimento? E qual a duração do incumprimento?
- Quantas entidades têm redução no valor da propina?
- Qual a demografia das entidades que geram receita por via do pagamento de juros?
- Quais os meios de pagamento mais utilizados pelos estudantes?
- Qual o valor resultante das prestações de propina arrecadado? E como tem evoluído ao longo do tempo?
- Quais os postos de recebimento mais utilizados pelos estudantes? E em que período?

A partir das questões suprarreferidas foram criados os seguintes indicadores: número de entidades em incumprimento, valor de propina pago por entidade e número de transações por meio de pagamento. Estes indicadores são importantes para o desenvolvimento da *data mart*, embora não sejam condição única. É também necessário desenvolver um método de extração, transformação e carregamento dos dados dos sistemas operacionais, o desenho e a implementação do modelo em estrela, a definição de vários cubos *OLAP* e a apresentação dos resultados, numa página *web,* sob a forma de *dashboards*.

Por fim, uma *data mart* oferece funcionalidades de *drill-down*, *roll-up*, *slice*, *dice* e *drill-across*, o que torna a apresentação dos resultados mais interessante do ponto de vista de quem se vai basear neles para tomar decisões, uma vez que será possível ver toda uma análise por unidades orgânicas e cursos. Permite, ainda, a junção de várias tabelas de factos para a apresentação de resultados, possibilitando a comparação de dois indicadores distintos.

Os processos *drill-down* e *roll-up* permitem ao utilizador aumentar, no primeiro caso, ou diminuir, no segundo, o nível de detalhe dos dados. Por exemplo, se estivermos a visualizar o valor de propina pago na Universidade, efetuando *drill-down*, os dados poderão ser apresentados por unidade orgânica, departamentos ou cursos. Pelo contrário, o *roll-up* faz o processo inverso, sendo, assim, os dados consolidados em níveis superiores de informação.

Slice é um processo usado para excluir os dados que não são necessários para a análise que se estiver a efetuar. Por exemplo, se estivermos interessados na análise do valor de propina pago na Universidade, considerando o ciclo de estudos, então, aplicando o processo *slice*, selecionamos a dimensão relativa ao ciclo de estudos escolhido, obtendo, desta forma, um subconjunto dos dados originais, relevante para a análise.

Dice é um processo semelhante ao anterior, diferindo apenas no número de dimensões que possibilita selecionar, por exemplo ciclo de estudos e prestação de propina.

Um *drill-across* é o processo que liga uma ou mais tabelas de facto com a mesma granularidade e com o mesmo conjunto de dimensões. Desta forma, é possível comparar dois indicadores distintos, por exemplo, o valor pago das propinas com o valor devido, podendo as operações de *drill-down*, *roll-up*, *slice* e *dice* serem, depois, aplicadas.

Dado que se trata de um projeto de desenvolvimento de *software*, fazem também parte dos objetivos do estágio tarefas como prototipagem, definição de requisitos, especificação e desenvolvimento de uma arquitetura, implementação, testes e validação de processos de *ETL* e dos *dashboards*.

1.2. Estrutura do Relatório

Nos capítulos seguintes, serão apresentadas todas as fases de desenvolvimento de uma *data mart*. Assim, no próximo capítulo, será possível encontrar o estado da arte, onde está descrito o processo implementado, atualmente, na UC, para a obtenção dos indicadores relacionados com o campo de ação deste estágio.

No terceiro capítulo, será possível encontrar toda a metodologia usada para o levantamento, definição e validação dos requisitos do projeto. Todos os requisitos, funcionais e não funcionais, se encontram aqui descritos e identificados.

No quarto capítulo, apresenta-se a arquitetura implementada, bem como os modelos de dados a usar e uma descrição do processo de *ETL*. É ainda possível encontrar um estudo das tecnologias existentes, presentemente, no mercado, para cada uma das partes constituintes da arquitetura.

No quinto capítulo, é apresentado o método de implementação do projeto e todos os detalhes e desafios, bem como o modo como foram superados.

No sexto capítulo, é mostrado o modo como a implementação foi testada e validada.

No sétimo capítulo, é exposta a metodologia de desenvolvimento adotada. Encontram-se ainda descritas as tarefas realizadas, os atrasos que houve, em relação ao plano inicial, e as medidas corretivas que foram aplicadas.

No oitavo e último capítulo, são apresentadas as conclusões gerais sobre o trabalho realizado.

Capítulo 2
Estado da Arte

Neste capítulo, é apresentada a situação atual da Universidade para o cálculo de indicadores sobre a receita de propinas e respetivos juros e de emolumentos, provenientes das entidades pagadoras, sejam elas estudantes ou entidades terceiras.

2.1. Situação Atual da UC

A Universidade dispõe, atualmente, de um sistema SAP para o qual foi adquirido um módulo que permitiria calcular um conjunto de indicadores, sem a necessidade de alocar recursos humanos. Hoje em dia, este sistema não é utilizado por nenhum colaborador da Universidade nas suas tarefas diárias de gestão. Por essa razão, cada vez que é necessário calcular indicadores de desempenho, é indispensável requisitar diversos funcionários de vários departamentos para calcular e validar os resultados para os indicadores requisitados. Desta forma, todos os indicadores relevantes para o desenvolvimento deste estágio foram obtidos, manualmente, pelos funcionários designados para o efeito, a partir do Nónio.

2.2. Estudo de Soluções Existentes no Mercado

Apesar de se ter optado por desenvolver *software* à medida, foi feito um estudo das soluções existentes no mercado, com a finalidade de perceber a oferta que existe atualmente. Esta análise permitiu compreender quais os pontos fortes e quais os pontos fracos de cada produto, conforme se pode constatar nos pontos seguintes.

Oracle Business Analytics

Este produto[1] é desenvolvido pela Oracle e resulta da conjugação de vários produtos da empresa. Assim, e dependendo das funcionalidades que o utilizador deseje, o custo de aquisição e de manutenção pode ser elevado. Fornece, ainda, um módulo específico para desenhar soluções de *Business Intelligence* para clientes *mobile*.

Style Intelligence

Este produto[2] é desenvolvido pela InetSoft e é bastante completo, ao nível da análise *OLAP*. Fornece ferramentas simples de utilizar para a produção de relatórios e *dashboards,* com um aspeto gráfico bastante apelativo. É um produto que tem conectores para os mais diversos sistemas de base de dados, no entanto não possui qualquer mecanismo para a tarefa de *ETL*. Fornece, igualmente, ferramentas para o desenvolvimento de *dashboards* e de relatórios específicos para clientes *mobile* e promete, ainda, ser um produto escalável, quando usado num *cluster*. É um produto pago, todavia disponibiliza uma versão para testes, totalmente funcional, durante cinco dias.

SiSence Prism

Este é um produto,[3] criado pela SiSense, que permite desenvolver *dashboards* de forma simples, usando um mecanismo de *drag-and-drop*. Os *dashboards* gerados têm um aspeto gráfico simples e, ao mesmo tempo, apelativo. Permite também desenhar processos de *ETL* de forma gráfica e suporta conectores, para as bases de dados mais populares, bem como *ERPs* e ficheiros de *Office* e *CSV*. Com este produto, é ainda possível integrar os *dashboards,* desenvolvidos num qualquer *website* já existente, contudo não suporta a definição de cubos *OLAP* e tem custos de aquisição.

SE BI

É um produto,[4] da SoftExpert, que permite desenvolver *dashboards* e relatórios numa página *web*. Possui conectores para os sistemas *ERP* e para aos motores de base de dados que reunem um maior número de utilizadores, no entanto não apresenta uma ferramenta gráfica apelativa, para a implementação do processo de *ETL*. É um produto que tem custos de aquisição e não permite a definição de cubos *OLAP*. Não obstante, permite a exportação dos gráficos, presentes nos *dashboards,* para *Excel* ou *PGN*.

MicroStrategy Analytics

Este produto é desenvolvido[5] pela MicroStrategy e tem várias versões disponíveis. Existem as versões *Analytics Desktop*, *Analytics Express* e *Analytics Enterprise*. Incidir-se-á a análise sobre a última versão, visto ser a mais completa. Neste sentido, destaca-se o elevado número de sistemas fonte que são suportados. Fornece conectores para todas as bases de dados relacionais, orientadas a colunas e *map-reduce* e oferece conectores para sistemas *ERP* e sistemas baseados em *SaaS*. Relativamente à análise *OLAP,* é possivel desenvolver *dashboards* e relatórios gráficamente apelativos, quer para *web*, quer para clientes *mobile,* e permite, ainda, a integração destes *dashboards* com o *Office*. Permite, além disso, exportar os *dashboards* e os relatórios para formatos de imagem e *PDF* e existe também suporte para colocar o produto em execução num *cluster*. Não permite a parametrização de cubos *OLAP* e é um produto pago.

SAP Business Intelligence

É desenvolvido pela SAP[6] e permite a criação de relatórios e *dashboards* para a *web*. É possível definir um conjunto de indicadores a monitorizar, independentemente dos dados estarem presentes no sistema ou noutro sistema externo. Suporta a distribuição da informação, através de mecanismos de *publish-subscribe,* o que permite que os utilizadores recebam a informação de forma rápida e simples. O processo de *ETL* não tem uma interface gráfica para a sua definição e não parece ser um produto com uma interface *web* apelativa. É um produto com custos de aquisição e de manutenção, mas permite o acesso aos relatórios e aos *dashboards* definidos, através de clientes *mobile*.

JasperSoft

É um produto[7] desenvolvido pale empresa Jaspersoft Corporation. Apresenta uma *interface* agradável e permite desenvolver *dashboards* e relatórios com os mais variados tipos de gráficos. Tem ainda uma componente de *ETL* que admite definir as transformações de forma gráfica e a conexão com vários sistemas *RDBMS* e *Non-RDBMS*. Possibilita, ainda, o

acesso, através de clientes *mobile,* e tem uma *API* que permite ao produto ser integrado noutro. Porém, é um produto que tem custos de aquisição.

De seguida, apresenta-se uma tabela que tem como objetivo resumir o estudo acima exposto.

Produtos	Suporta desenho de processos *ETL*	Permite criar relatórios	Permite criar *dashboards*	Permite exportar os dados	Necessário obter licenças
Oracle Business Analytics	✓	✓	✓	✗	✓
Style Intelligence	✗	✓	✓	✗	✓
SiSence Prism	✓	✓	✓	✓	✓
SE-BI	✓	✓	✓	✓	✓
MicroStrategy Analysis	✗	✓	✓	✓	✓
SAP Business Objects	✓	✓	✓	✗	✓
JaperSoft	✓	✓	✗	✓	✓

Tabela 1 - Quadro comparativo *software business intelligence.*

2.3. Sumário

Apesar das soluções apresentadas terem popularidade e sucesso no mercado de *Business Intelligence*, elas têm de ser adaptadas a cada instituição, isto é, apesar das soluções existentes tratarem das várias fases de implementação de uma *data warehouse*, é sempre necessário adaptar as fórmulas de cálculo dos indicadores à realidade da instituição, especificar indicadores que não estão contemplados na solução original, definir as fontes de dados a utilizar e o modo como os indicadores serão apresentados, bem como todo o processo de *ETL*. Para além disto, é necessário pagar uma licença para a sua utilização que, em alguns casos, pode ter de ser paga anualmente. Dada a situação económica atual da Universidade e a existência de conhecimento sobre como implementar este tipo de sistemas, foi decidido construir uma solução à medida.

Os factores diferenciadores deste projeto, em relação ao modo como os indicadores são obtidos atualmente na UC, são a rapidez com que os indicadores vão poder passar a ser obtidos, isto é, em vez do atual tempo de espera, estes encontrar-se-ão disponíveis, numa página *web*, quase em tempo real. Passam, além disso, pela variedade de análise que pode ser feita instantaneamente, dado que, em vez de os indicadores serem apresentados em papel, e, serem, portanto, estáticos, são apresentados em *dashboards* dinâmicos, que permitem a análise dos valores de receita de propinas, de juros, de emolumentos e a utilização dos meios de pagamento, sob vários pontos de vista diferentes, rapidamente. Por último, como se trata de *dashboards* dinâmicos, os resultados da análise passam a ser visualizados em gráficos interativos, mais apelativos que as tradicionais tabelas.

Capítulo 3
Processo de Identificação de Requisitos

Nesta secção, é exposto todo o processo de identificação de requisitos e apresentados, de forma sucinta, os requisitos identificados.

3.1. Identificação de Requisitos

Para se proceder à identificação dos requisitos, a abordagem passou por reunir com os principais mentores e utilizadores finais do projeto. Neste âmbito, destacamos, em primeiro lugar, uma reunião com o reitor da Universidade, Professor Doutor João Gabriel Silva, para definir os objetivos gerais a incluir no sistema a desenvolver. Os indicadores do negócio foram também pesquisados no plano estratégico e de ação[8] e no relatório de contas[9] da Universidade.

Após esta primeira identificação de requisitos, foi adotada uma abordagem em que se desenvolveu um protótipo rápido, com o objetivo de mostrar aos utilizadores finais as funcionalidades que se pretendem implementar e o modo como estarão disponíveis para poderem ser usadas, dissipando, desta maneira, qualquer falha na comunicação entre as partes.

De seguida, o protótipo rápido foi validado por ambas as partes, com o objetivo das funcionalidades ficarem definidas formalmente. Ao longo de todo este processo de desenvolvimento e validação, foram tidas reuniões com os elementos da equipa reitoral, com o reitor e com a vice-reitora, Professora Doutora Margarida Mano.

Depois da melhoria de algumas funcionalidades, foi realizada uma última reunião para validação de todos os módulos do projeto, onde foram totalmente definidos e aprovados as funcionalidades e os indicadores a serem abordados em cada módulo. Estiveram presentes o Professor Doutor Bruno Cabral, na qualidade de orientador do estágio, o Engenheiro Pedro Pinto, como orientador do módulo sucesso escolar, os colegas responsáveis pelo desenvolvimento dos restantes módulos do projeto, o Chefe de Divisão da Divisão de Planeamento, Gestão e Desenvolvimento, Filipe Rocha, e ainda um consultor externo.

Após esta reunião de validação, foi feita a especificação formal dos requisitos funcionais e não funcionais do sistema. Para a especificação formal de todos estes requisitos, adotou-se a seguinte estratégia: os requisitos funcionais foram especificados através das funcionalidades criadas no protótipo apresentado e validado anteriormente. Como se trata de um projeto constituído por vários módulos, houve a necessidade de integração de alguns dos requisitos funcionais comuns aos restantes módulos do projeto, embora o sistema final a ser desenvolvido tenha de ser apenas um só. Assim, o código com a expressão RF_GE_XX representa os requisitos funcionais comuns a todos os módulos da *data warehouse*. O código RF_XX representa todos os requisitos funcionais específicos deste módulo.

3.2.Requisitos Funcionais

Na tabela seguinte, é apresentada uma descrição sucinta dos requisitos funcionais identificados, bem como dos indicadores que lhes dão origem. É de notar que em engenharia de *software* é comum ser atribuído um grau de importância aos requisitos identificados. Portanto, os requisitos, cuja importância é elevada, são os que têm

necessariamente de constar no sistema, uma vez que a falta destes compromete as funcionalidades necessárias ao seu funcionamento. Por outro lado, os requisitos, cuja importância é média, são aqueles que vêm aprimorar e introduzir funcionalidades que melhoram a qualidade global do sistema e, caso não estejam contemplados no sistema final, não comprometem as funcionalidades chave.

Código	Designação	Prioridade	Descrição Sucinta
RF_GE_01	Autenticação	Elevada	Permitir realizar o *login* com as credenciais da UC e o controlo de acesso por módulos.
RF_GE_02	Fechar sessão	Elevada	Permitir ao utilizador fazer *logout*.
RF_GE_03	Término de sessão	Elevada	Terminar sessão, após tempo de inatividade.
RF_GE_04	Navegação entre módulos	Média	Permitir a troca entre módulos a qualquer momento.
RF_GE_05	Navegação interna	Elevada	Permitir ao utilizador subir e descer na granularidade.
RF_GE_06	Parâmetros gerais	Elevada	Mostrar e selecionar as dimensões para fazer *slice*.
RF_GE_07	Parâmetros de tempo	Elevada	Mostrar a dimensão temporal e os vários modos disponíveis para seleção do tempo.
RF_GE_08	Esconder parâmetros	Baixa	Esconder o menu lateral.
RF_GE_09	Seção de ajuda	Elevada	Disponibilizar uma secção de ajuda.
RF_GE_10	Informação auxiliar	Elevada	Informação sobre a análise apresentada nos gráficos.
RF_GE_11	Visualização: gráfico ⇔ tabela	Elevada	Mostrar a tabela que dá origem ao gráfico.
RF_GE_12	Exportar informação na tabela	Baixa	Exportar as tabelas que dão origem aos gráficos para *Excel*.
RF_13	Visão geral das fontes de receita	Elevada	Deve ser possível mostrar a evolução dos valores das várias fontes de receita da UC.
RF_14	Análise das transações e montantes por meio de pagamento	Elevada	Deve ser possível ver quais os meios de pagamento e os postos de pagamento mais utilizados pelos estudantes.

RF_15	Análise do valor de juros pago	Elevada	Deve ser possível ver o valor pago de juros pelos estudantes e ainda efetuar *drill-down* e *roll-up*.
RF_16	Análise do valor médio de juros pago	Elevada	Deve ser possível ver o valor médio de juros pago. Deve ainda ser possível fazer *drill-down* e *roll-up*.
RF_17	Análise do valor pago de propina	Elevada	Deve ser possível ver o valor de propina pago pelos alunos. É imperativo haver *drill-down* e *roll-up*.
RF_18	Análise do valor reduzido de propina	Elevada	Deve ser possível ver o valor de propina reduzido, em relação ao valor do plano de estudos. Deve ainda ser possível quantificar os estudantes que têm redução do valor de propina. *Drill-down* e *roll-up* são funcionalidades que devem estar disponíveis.
RF_19	Análise de entidades em incumprimento no pagamento de propina	Elevada	Deve ser possível ver o número de entidades que se encontram em incumprimento. Funcionalidades de *drill-down* e *roll-up* devem estar presentes.
RF_20	Análise do tempo de pagamento de propina	Elevada	Deve ser possível mensurar, num intervalo discreto, o tempo que as entidades demoram a pagar a propina devida. Deve ser possível efetuar *drill-down* e *roll-up*.
RF_21	Análise do valor de propina de cobrança duvidosa	Elevada	Deve ser possível determinar, num intervalo discreto, os valores de cobrança duvidosa. Deve ser possível efetuar *drill-down* e *roll-up*.
RF_22	Análise da taxa de incobrabilidade	Elevada	Deve ser possível calcular, num intervalo discreto, o valor da taxa de incobrabilidade. Deve ser possível efetuar *drill-down* e *roll-up*.
RF_23	Discriminação da receita mensal por anos letivos	Elevada	Deve ser possível apurar, num intervalo mensal, trimestral, ou semestral, a receita do ano civil, discriminada por ano letivo. Deve ser possível efetuar *drill-down* e *roll-up*.

Tabela 2 – Descrição sucinta dos requisitos funcionais.

3.3. Requisitos Não Funcionais

Como existe a necessidade de integrar os vários módulos num só sistema, os requisitos não funcionais são, por essa razão, transversais a todos os módulos do projeto. Consequentemente, a sua identificação foi realizada em conjunto com os colegas encarregados do desenvolvimento dos módulos do sucesso escolar, dos recursos humanos e do custo com o ensino.

Nas tabelas subsequentes, os requisitos não funcionais são apresentados de forma sucinta e de acordo com a respetiva importância para o sistema final.

Código	Designação	Prioridade	Descrição Sucinta
RFN_S_01	Atualização de dados	Elevada	Processo de *ETL* e atualização do cubo *OLAP* devem ser autónomos.
RFN_S_02	Compatibilidade (*browser*)	Elevada	A aplicação deve ser compatível com os *browsers* mais modernos.
RFN_S_03	Compatibilidade (SO)	Média	A aplicação deve ser compatível com sistemas *Unix* e *Windows*.

Tabela 3 – Descrição sucinta dos requisitos de suporte.

Código	Designação	Prioridade	Descrição Sucinta
RNF_O_01	*Hardware*	Elevada	As máquinas a utilizar devem ter, pelo menos, 4*GB* de *RAM,* 250*GB* de espaço em disco e um processador *dual-core*.

Tabela 4 – Descrição sucinta dos requisitos da categoria outros.

3.4. Sumário

A metodologia de trabalho aplicada centrou-se na leitura dos documentos plano estratégico e de ação e relatório de contas da Universidade e nas reuniões com os elementos da equipa reitoral para a definição de indicadores e funcionalidades gerais do sistema a desenvolver. Todas as funcionalidades apresentadas nas tabelas, exceto nas números 3 e 4, foram analisadas e validadas pelos utilizadores finais do projeto, tendo como suporte o protótipo rápido desenvolvido para ilustrar as funcionalidades do sistema. Após a validação das funcionalidades, foi feita a escrita formal dos requisitos do sistema.

Uma descrição mais completa de todos os requisitos aqui apresentados pode ser encontrada no anexo 1, bem como toda a documentação relacionada com os protótipos desenvolvidos e aprovados. É, ainda, possível encontrar, no referido anexo, todos os indicadores identificados na área de negócio do presente módulo, tal como todos os documentos, elaborados conjuntamente com os colegas responsáveis pelo desenvolvimento dos restantes módulos, que pretendem definir regras gerais de especificação de funcionalidades comuns a todos os módulos.

Capítulo 4
Arquitetura

Neste capítulo, é apresentada a arquitetura do sistema a implementar, são explicados todos os processos que a constituem e os modelos de dados, tal como o plano de *ETL*. É, igualmente, apresentada uma análise das tecnologias a utilizar em cada processo.

4.1. Arquitetura Geral

Na figura 1, é apresentada a arquitetura geral, desenvolvida durante o estágio.

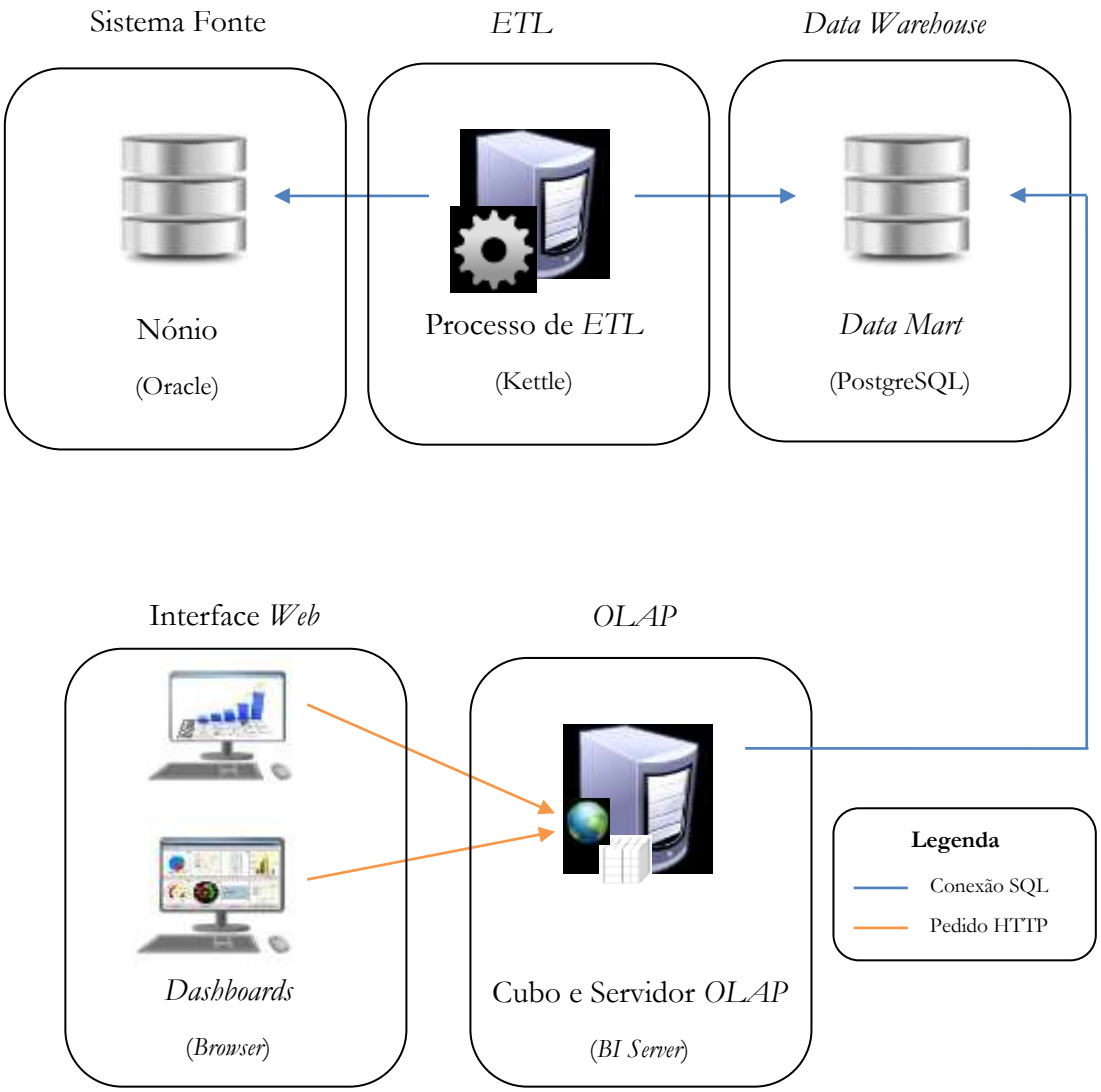

Figura 1 – Especificação da arquitetura do sistema.

Como facilmente se depreende da imagem, existe um sistema fonte que alimenta todo este módulo. O sistema expõe um conjunto de vistas sobre as quais o processo de *ETL* irá fazer a extração dos dados. Após a extração, será feita a transformação e o carregamento para a *data mart*. Concluído o carregamento, serão configurados os vários cubos, respeitantes aos indicadores definidos no capítulo anterior, que, por sua vez, alimentarão os *dashboards* que serão mostrados aos utilizadores finais.

De seguida, vão ser detalhados todos os componentes e a sua interação com os que lhes estão subjacentes e explicados todos os conceitos técnicos relevantes.

4.2. Sistema Fonte

O único sistema fonte para este projeto é o Nónio, a partir do qual foram criadas e expostas vistas, com os dados que serão utilizados no módulo encarregado pela execução dos processos de *ETL*. Nas vistas, existe informação sobre o pagamento de propinas, sobre emolumentos e sobre o percurso escolar dos alunos. Como há a necessidade de relacionar informação sobre os pagamentos efetuados pelos alunos com os seus dados curriculares e das parcelas de prestações de propinas com entidades terceiras, em todas as vistas, existem chaves que permitem o seu relacionamento.

As vistas expostas no Nónio são as seguintes:

- MVIEW_CURSOS_UO_A
- MVIEW_CURSOS_UO_PB
- MVIEW_DADOS_FIN
- MVIEW_DEMOGRAFIA_MATRICULA
- MVIEW_INSCRICOES_CURSO
- MVIEW_SITUACOES_ESPECIAIS
- MVIEW_UNIDADES_ORGANICAS

Nas duas primeiras vistas, encontram-se informações como o nome do curso, a sua sigla, o seu grau e categoria e a unidade orgânica responsável por ele, num determinado ano letivo. Ambas as vistas têm as mesmas colunas, no entanto, na primeira vista, encontram-se os cursos pós-Bolonha e, na segunda, encontram-se os cursos pré-Bolonha. A informação foi separada em duas vistas, porque a vista MVIEW_CURSOS_UO_A é partilhada com os restantes módulos.

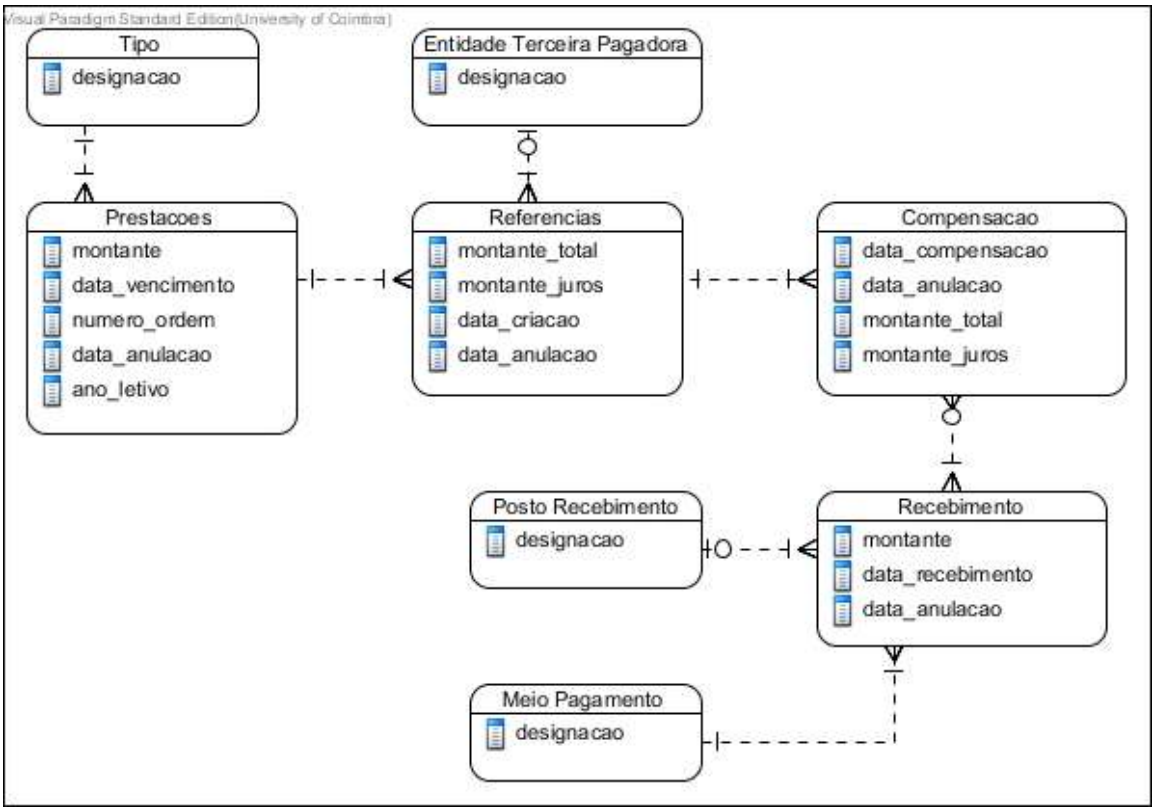

Figura 2 - Modelo simplificado de dados da vista MVIEW_DADOS_FIN.

Como é mostrado na figura 2, uma prestação tem um tipo associado. Esta entidade diz se a prestação é propina, certidão ou um outro emolumento. Não se encontra representada no diagrama, mas a entidade prestação tem uma relação de M:1 com uma entidade matrícula. É a partir desta última entidade que, depois, se consegue fazer a relação entre prestação e curso.

Dado que uma prestação de propina pode ser dividida em várias parcelas e cada parcela pode ser paga por uma entidade pagadora diferente, esta informação é registada na entidade referência. Como é nesta entidade que se encontram os registos com a granularidade mais fina em relação à dívida, será esta entidade a ser utilizada para a sua monitorização, tal como se mostra nos capítulos subsequentes.

A entidade recebimento destina-se a registar todos os recebimentos efetuados pelos estudantes, sejam eles feitos no *ATM* ou nos serviços académicos. A entidade posto de recebimento identifica a localização do serviço académico utilizado pelo aluno e a entidade meio de pagamento identifica o meio de pagamento utilizado.

Visto que existe uma relação de M:N entre a entidade recebimento e a entidade compensação, há, no modelo físico, uma tabela de nome parcelas de compensação que contém, como o nome indica, as parcelas que fazem parte de uma compensação. É nesta tabela que se encontra a granularidade mais fina, ao nível da receita, e, a partir desta tabela, será feita a sua monitorização, como será explicado nos capítulos posteriores.

Na vista MVIEW_DADOS_FIN, é onde se encontram as informações relativas ao pagamento de propinas e emolumentos. Cada linha desta vista representa uma parcela de compensação associada a uma referência, tal como se pode ver no diagrama ER da figura 2.

As restantes vistas, à exceção da última, contêm um registo por matrícula com a informação sobre demografia, regime de inscrição e situações especiais, respetivamente. Na vista das unidades orgânicas, cada registo tem as unidades orgânicas existentes na UC, bem como, caso exista, a unidade orgânica superior responsável por ela.

4.3. *Data mart*

É neste módulo que todos os dados serão guardados durante todo o tempo de vida da *data mart*. É, por isso, necessário desenvolver um modelo de dados que permita, por um lado, suportar uma grande quantidade de dados, durante anos, e, por outro, que pesquisas sobre ele efetuadas sejam rápidas a devolver resultados. Considerando as premissas apresentadas anteriormente e tendo em consideração o livro *The Data Warehouse Toolkit. The complete guide to dimensional modelling*[10] de Ralph Kimball, o modelo de dados adoptado é o modelo em estrela. Para além deste modelo, é, ainda, necessário desenvolver uma área temporária para guardar os dados extraídos dos sistemas fontes. Nas subsecções seguintes, ambos os modelos serão explicados em detalhe.

Modelo em estrela

Este modelo é composto por tabelas de facto e por dimensões. As tabelas de facto são tabelas com chaves estrangeiras e factos, isto é, valores numéricos. As chaves estrangeiras, presentes nestas tabelas, são necessárias para referenciar as tabelas de dimensão, ou seja, tabelas que pretendem descrever os factos capturados. Os factos pretendem capturar eventos e são importantes para mensurar o desempenho do estado do negócio, através do cálculo de indicadores de *performance*.

No contexto deste módulo, os factos capturados são, regra geral, aditivos, isto é, a soma e a média são operações cujo resultado, no eixo temporal, é concordante com a análise que está a ser realizada. No entanto, existem tabelas de factos, onde os factos medidos podem ser valores semi-aditivos, isto é, a sua soma, no eixo temporal, deixa de fazer sentido, apenas a sua média é relevante para a análise em questão.

As dimensões são tabelas, compostas por vários atributos, que descrevem os factos presentes nas tabelas de facto. Cada dimensão tem uma chave primária que a liga à tabela de factos. A relação entre as tabelas de facto e as dimensões são do tipo N:1, ou seja, cada registo na tabela de factos só se pode ligar a um registo nas tabelas de dimensão, ao invés, cada registo nas tabelas de dimensão pode ser referenciado em vários registos nas tabelas de facto.

Este modelo contrasta com os modelos que, habitualmente, são desenhados para responder a necessidades operacionais do negócio, dado que a informação, presente num modelo em estrela, está agrupada de acordo com relações lógicas entre os dados, o que torna mais fácil compreender a informação e navegar pela sua estrutura. Por outro lado, a estrutura do modelo em estrela faz com que as chaves forasteiras, existentes nas tabelas de facto, sejam legítimas, estando relações de integridade entre dimensões e tabelas de factos garantidas, e, consequentemente, não ser necessário garantir estas restrições no motor de base de dados, no momento da inserção dos registos. Esta garantia é dada pelo processo de *ETL*, visto que, quando as tabelas de facto são carregadas, primeiro são obtidas as chaves primárias dos registos, presentes nas dimensões necessárias para o facto a ser carregado. É ainda possível referir que o modelo em estrela, relativamente ao modelo normalizado, permite a execução mais rápida e a simplificação de *queries*, uma vez que existe um pequeno número de dimensões com as quais é necessário fazer *joins*.

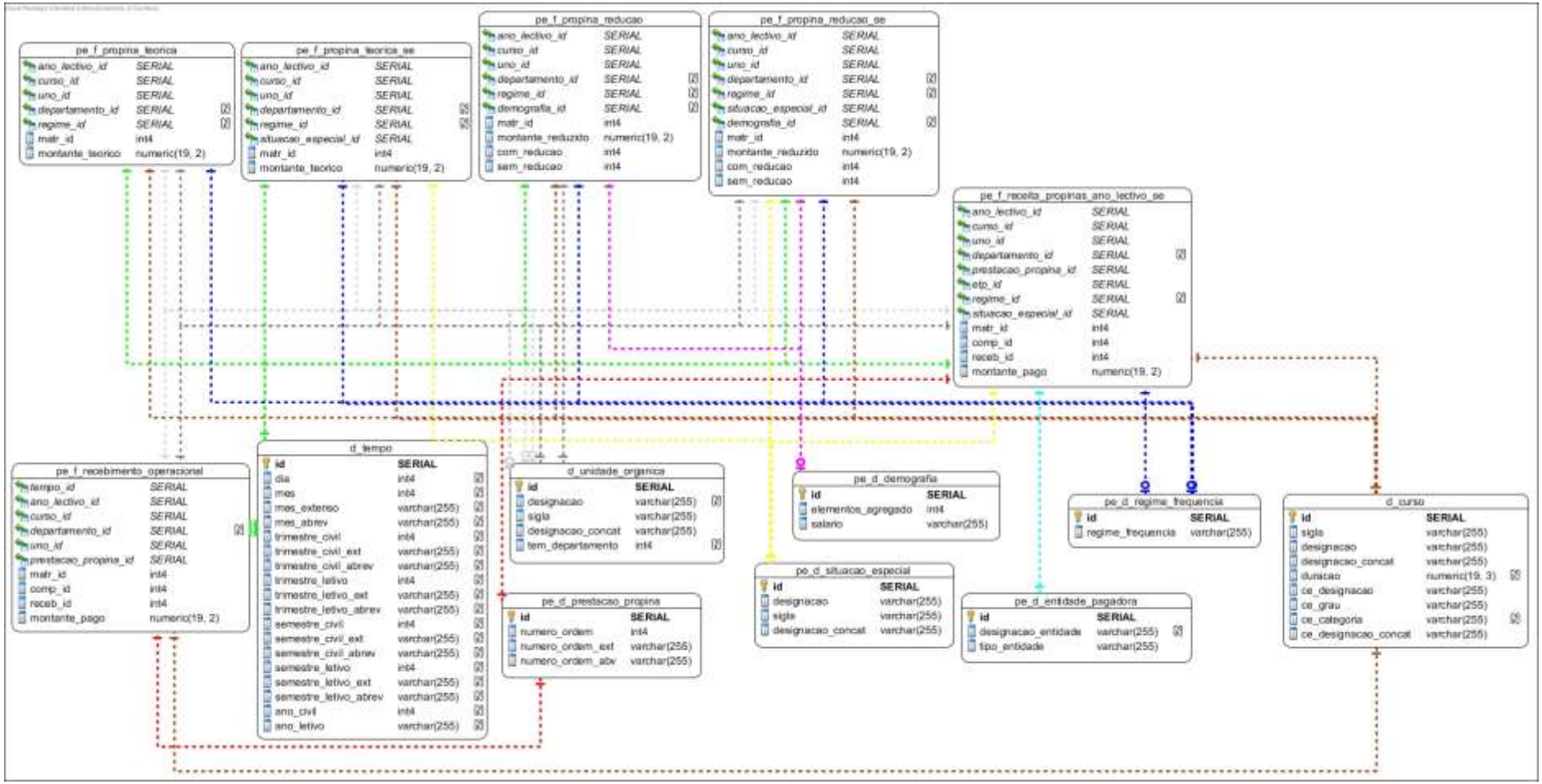

Figura 3 - Modelo de dados indicadores de propinas.

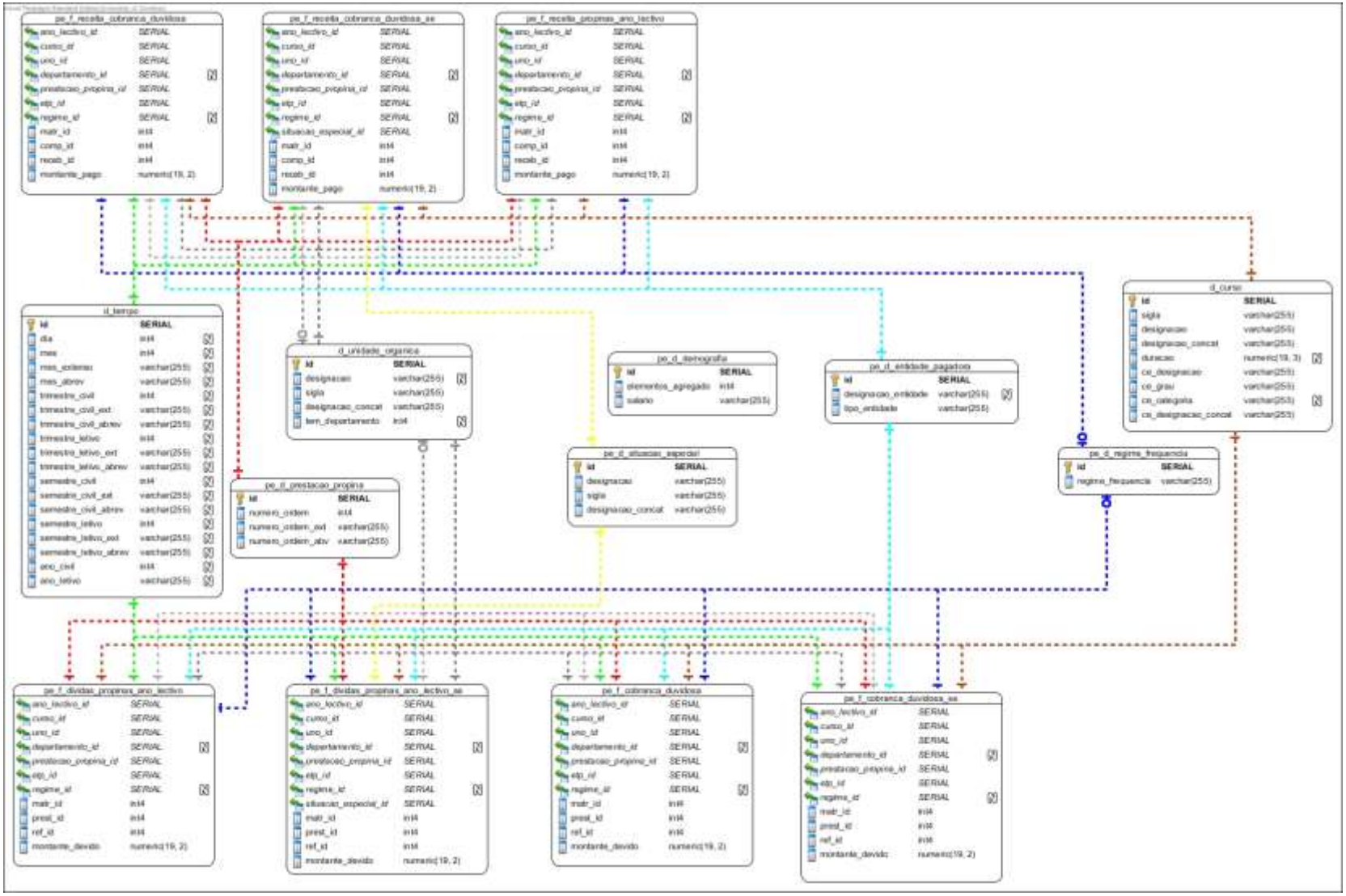

Figura 4 - Modelo de dados indicadores de propinas.

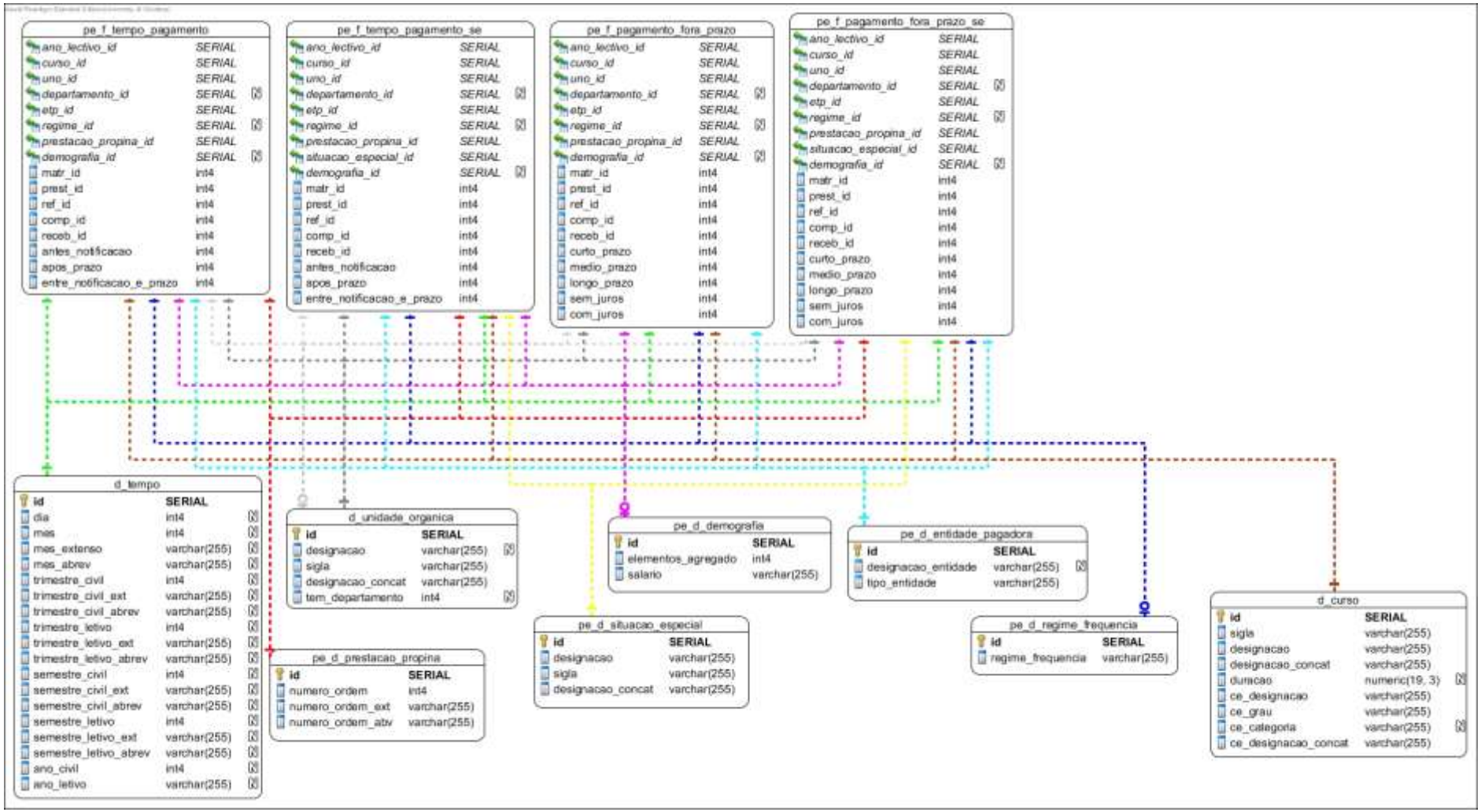

Figura 5 - Modelo de dados indicadores de propinas.

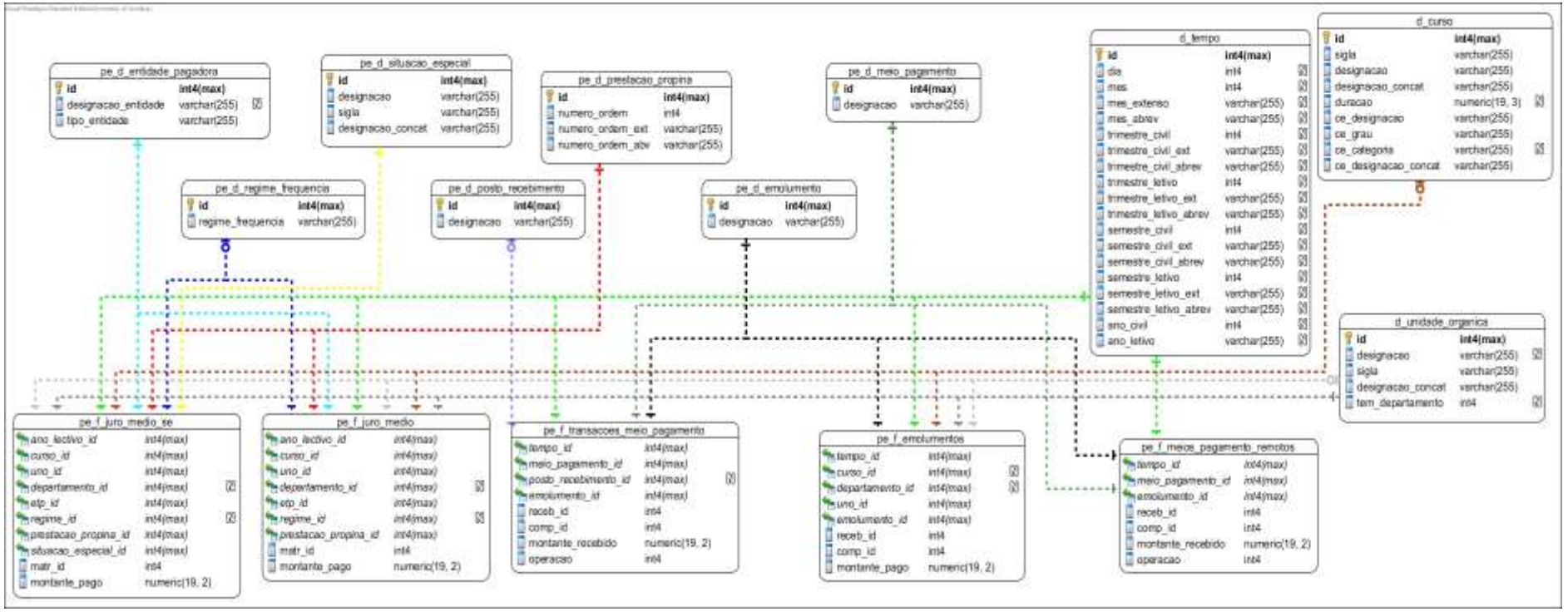

Figura 6 - Modelo de dados indicadores de juros, emolumentos e meios de pagamento.

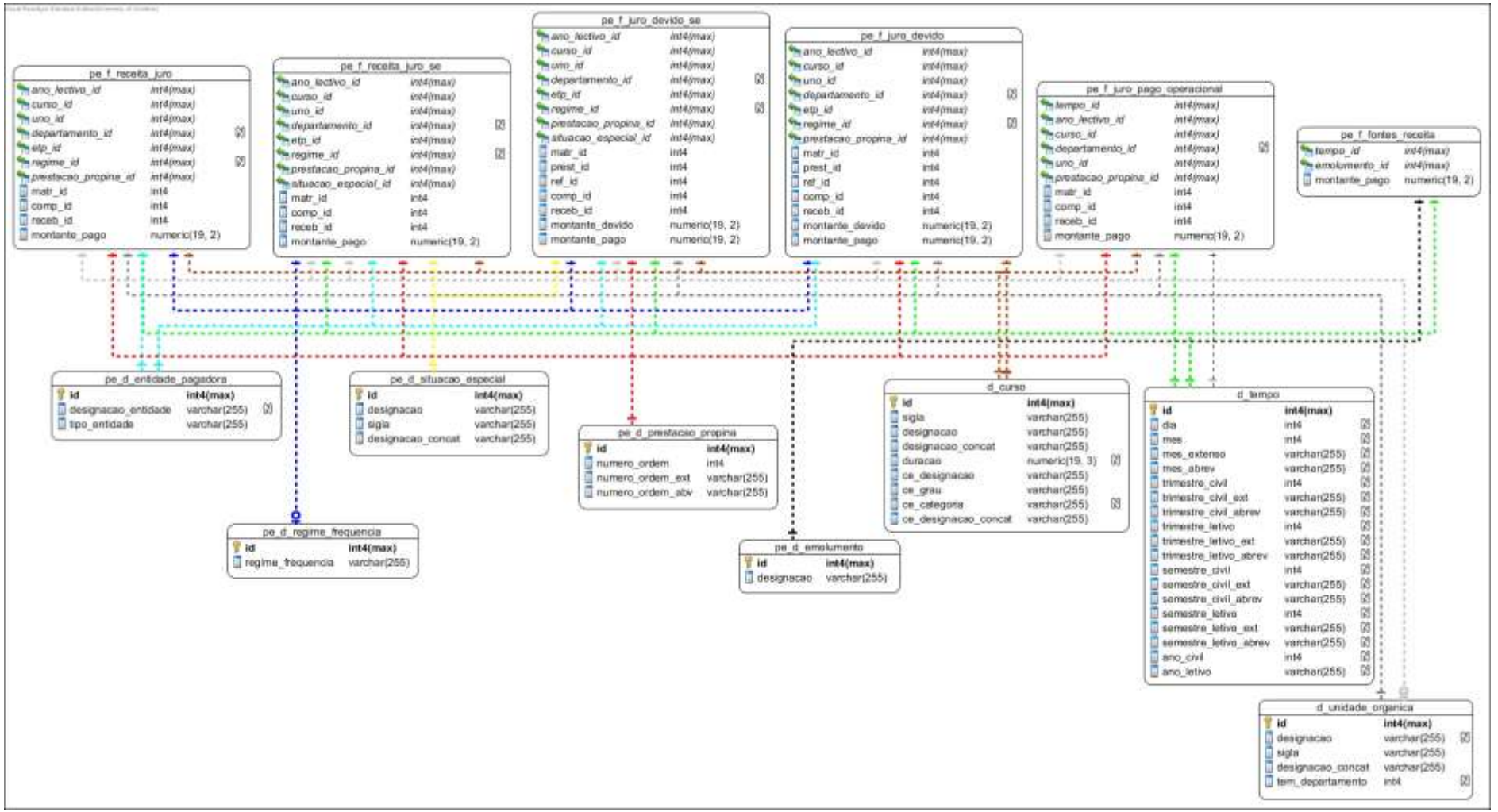

Figura 7 - Modelo de dados indicadores de juros, emolumentos e meios de pagamento.

As figuras 3 a 7 refletem o modelo em estrela, desenvolvido no âmbito deste estágio. Devido à sua grande extensão, o modelo foi seccionado em várias figuras. Nas figuras 3, 4 e 5, apresentam-se as tabelas de facto e as dimensões relativas aos indicadores da cobrança de propinas e, na figura 6 e 7, apresentam-se as tabelas de facto e as dimensões relativas aos indicadores da cobrança de juros e emolumentos e aos indicadores referentes aos meios de pagamento. É de notar que as dimensões que aparecem repetidas nos vários diagramas representam uma só. As figuras foram unicamente separadas para que fosse possível mostrar o modelo de dados de forma compreensível.

A partir do modelo de dados apresentado anteriormente, foi possível definir um conjunto de estrelas de forma concetual. Uma estrela é constituída por uma tabela de factos e as dimensões que a descrevem. Tomando o exemplo da tabela de factos pe_f_emolumentos, a estrela resultante conteria esta tabela de factos e as dimensões d_tempo, d_curso, d_unidade_organica e pe_d_emolumento. Geralmente, este conjunto de tabelas é arranjado de modo a parecer uma estrela, isto é, a tabela de factos é rodeada pelas dimensões, daí o seu nome.

Podem ser criados agregados para qualquer tabela de facto presente nas estrelas, de modo a permitir que a execução das *queries* sobre estas estruturas seja mais rápida. Deste modo, podem-se criar vários agregados, a partir de uma tabela de factos, bastando, para isso, fazer as combinações das colunas que contêm as chaves forasteiras das dimensões. Tomando como exemplo a tabela pe_f_emolumentos, que contém cinco dimensões, podem-se criar agregados, combinando essas cinco dimensões uma a uma, duas a duas, três a três e quatro a quatro. Dado o elevado número de tabelas e de dimensões presente em cada uma delas, criar todos estes agregados não era viável, tendo sido criados apenas alguns, como se pode ver no capítulo seguinte.

As tabelas de facto, que monitorizam dívida, têm granularidade referência (parcela de prestação de propina) e as tabelas de facto, que monitorizam receita, têm granularidade parcela de compensação (valores totais ou parciais correspondentes a determinada prestação). Assim, as tabelas pe_f_dividas_propinas_ano_letivo, pe_f_cobranca_duvidosa e pe_f_juro_devido têm granularidade referência. As restantes tabelas têm granularidade parcela de compensação, à exceção das tabelas pe_f_juro_medio, pe_f_propina_reducao e pe_f_propina_teorica que têm granularidade matrícula.

As tabelas com a terminação "se" têm granularidade referência por situação especial, ou parcela de compensação por situação especial. Exemplificando, dado que a tabela de factos pe_f_divida_propinas_ano_letivo tem granularidade referência, a tabela de factos pe_f_divida_propinas_ano_letivo_se terá granularidade referência por situação especial. Na prática, o que acontece nestas tabelas é que é inserido um registo da mesma referência ou parcela de compensação por cada situação especial que lhe está associada.

A partir do modelo em estrela, apresentado previamente, e da granularidade dos indicadores, explanada na tabela anterior, apresenta-se, na tabela seguinte, o espaço ocupado atualmente na *data mart* que foi medido com uma função do PostgreSQL que permite, entre outras informações, saber o espaço ocupado em disco por todos os registos existentes na tabela. É, ainda, apresentada uma estimativa a dez anos.

Tabela de Factos	Espaço Ocupado (*Bytes*)
pe_f_cobranca_duvidosa	6591994
pe_f_cobranca_duvidosa_se	456827
pe_f_dividas_propinas_ano_letivo	13458399
pe_f_dividas_propinas_ano_letivo_se	9845027
pe_f_emolumentos	3145550
pe_f_fontes_receita	27524
pe_f_juro_devido	2072126
pe_f_juro_devido_se	1474092
pe_f_juro_medio	945983
pe_f_juro_medio_se	766722
pe_f_juro_pago_operacional	1184123
pe_f_meio_pagamento_remoto	8504440
pe_f_pagamento_fora_prazo	4865400
pe_f_pagamento_fora_prazo_se	4726424
pe_f_propina_reducao	3020324
pe_f_propina_reducao_se	2314724
pe_f_propina_teorica	2380713
pe_f_propina_teorica_se	1822154
pe_f_recebimento_operacional	10927940
pe_f_receita_cobranca_duvidosa	10199040
pe_f_receita_cobranca_duvidosa_se	1968960
pe_f_receita_juros	1137633
pe_f_receita_juros_se	869544
pe_f_receita_propinas_ano_letivo	9280484
pe_f_receita_propinas_ano_letivo_se	10221254
pe_f_tempo_pagamento	11028332
pe_f_tempo_pagamento_se	7905016
pe_f_transacoes_meio_pagamento	10326220

d_tempo	5819246
d_curso	162790
d_unidade_organica	3088
pe_d_demografia	3165
pe_d_emolumento	3581
pe_d_entidade_pagadora	9604
pe_d_meio_pagamento	344
pe_d_posto_recebimento	381
pe_d_prestacao_propina	208
pe_d_regime_frequencia	72
pe_d_situação_especial	8042
TOTAL	147477490

Tabela 5 – Estimativa do tamanho da *data mart*.

Para realizar a estimativa a dez anos e assumindo que o número de registos se mantém constante, as tabelas de facto e dimensão da *data mart* irão ocupar cerca de 1,5 *Gigabytes*. Quero salientar que o espaço total, apresentado na tabela, e consequente estimativa apenas se referem às tabelas apresentadas no modelo de dados, não sendo considerado o espaço ocupado por índices, vistas materializadas e pelos vários agregados que cada tabela pode ter e pelos cubos que são especificados no *Mondrian*.

Área temporária

Na figura8, apresenta-se o modelo da área temporária. As colunas com cor azul representam colunas cuja informação é produzida aquando da sua extração.

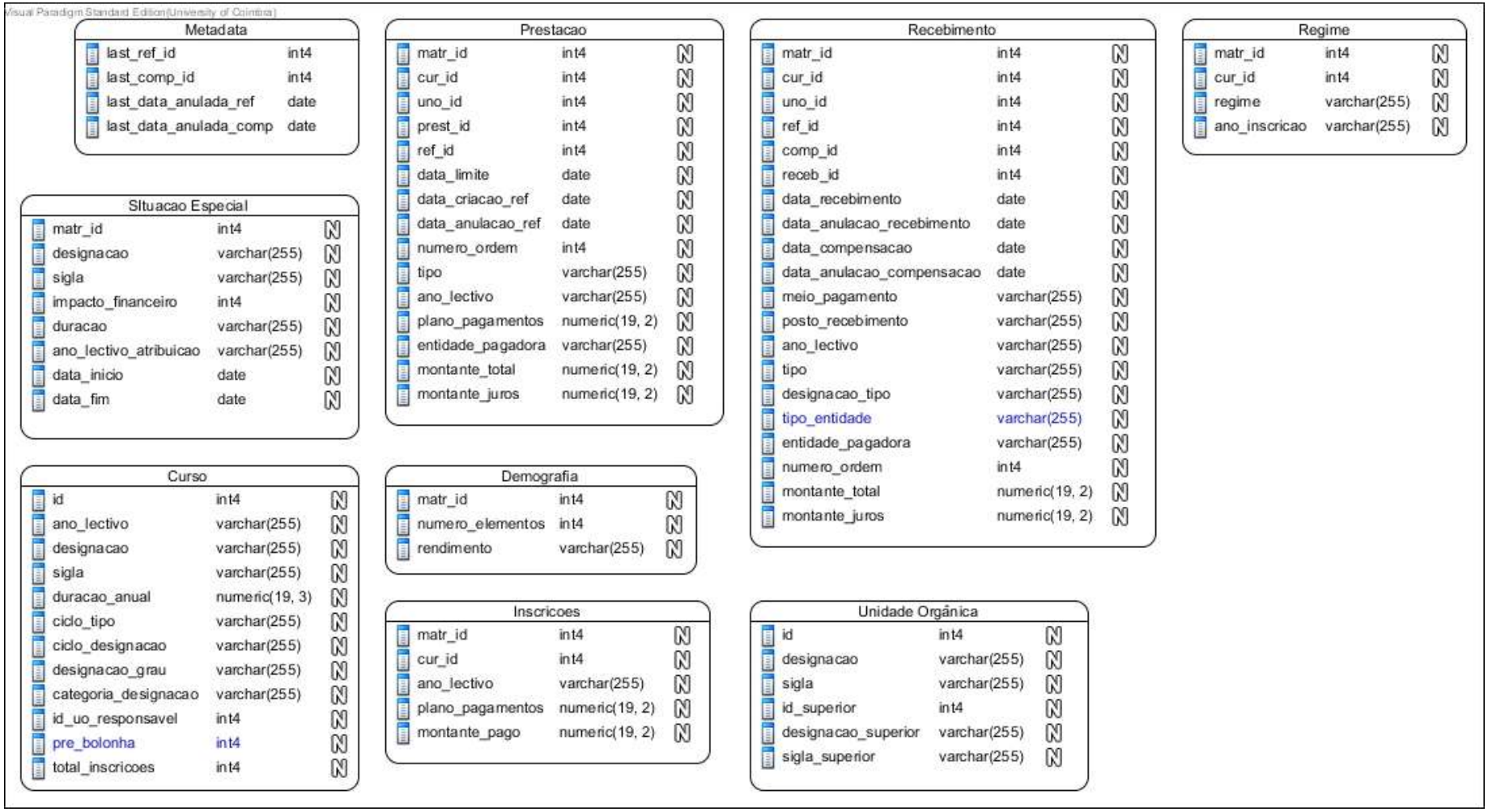

Figura 8 - Modelo de dados da área temporária

O modelo da área temporária, embora seja implementado numa base de dados, pouca relação tem com o modelo de dados, descrito na secção anterior, ou com o modelo de dados relacionais, usado em sistemas operacionais. Trata-se de um modelo desenhado especificamente para que o processo de *ETL* possa realizar a limpeza e transformação dos dados, sem interferir com os sistemas fonte. Para além de fornecer um espaço temporário para a armazenagem de dados antes do seu processamento, serve também para que seja mais fácil fazer a integração de dados, provenientes das várias vistas presentes no Nónio. Assim, este modelo deve ser desenhado de forma a garantir que os dados extraídos das vistas possam ser aqui guardados, temporariamente, para processamento futuro. Para além disso, deve ser, ainda, garantido que apenas o processo de *ETL* tem acesso aos dados presentes neste modelo, pois, como já foi referido anteriormente, trata-se de um modelo temporário, onde os dados são processados, que não serve, de maneira nenhuma, para o seu armazenamento futuro ou para a execução de *queries* destinadas a mostrar informação nos sistemas de análise.

4.4. Processo de Extração, Transformação e Carregamento

Este módulo é responsável pela obtenção dos dados do Nónio e pela sua colocação no modelo da área temporária.

Como os dados a transferir para a área temporária podem ser em número elevado (dados de propinas e respetivos juros, emolumentos e meios de pagamento, informação sobre os cursos da UC e as suas unidades orgânicas, inscrições dos alunos nos cursos, a sua demografia e situações especiais que lhes estejam atribuídas), é necessário que esta fase seja implementada de forma eficiente, uma vez que vai interagir com os serviços oferecidos à comunidade universitária, logo a sua interrupção ou degradação de *performance* podem ter consequências desastrosas. Uma vez carregados os dados de todas as vistas apresentadas na secção anterior para a área temporária, passamos à segunda fase do processo.

Numa segunda fase, serão realizadas todas as transformações necessárias aos dados para que seja possível popular o modelo em estrela, definido anteriormente e, assim, colocar a informação disponível para consulta. É nesta fase que reside a principal dificuldade de todo este processo, porque é aqui que os dados têm de ser limpos, ou seja, é necessário remover tudo o que possa ser considerado lixo, que esteja inconsistente, e que, em última análise, não seja necessário para preencher o modelo em estrela. Para além disso, é necessário tratar os dados, pois estes podem conter valores ou caracteres que tornariam os dados apresentados nos sistemas de análise incompreensíveis ou inconsistentes. Apesar de se conhecerem os dados fonte e de ser possível prever, com alguma exatidão, as operações de limpeza, de normalização, de correção de formatações e de inconsistências, por vezes, ou porque se mudaram as regras do negócio, ou porque existe algum caso menos comum no modelo de negócio, as transformações aplicadas aos dados podem não conseguir cobrir todas as situações, levando a que dados errados sejam carregados para o modelo em estrela que, posteriormente, alimentam os sistemas de análise. Esta fase reveste-se de grande importância, na medida em que todo este processo tem de ser automático e periódico no tempo. É, também, neste processo que todos os agregados são calculados, isto é, no caso de ser necessário calcular um valor médio ou a soma de vários valores, para mais tarde popular tabelas de factos no modelo em estrela, é aqui que estas transformações devem ser definidas.

Por fim, na fase de carregamento, após todos os dados limpos e, se necessário, agregados, são carregados para o modelo em estrela, a fim de serem consultáveis pelos sistemas de análise. Esta fase compreende a passagem dos dados da área temporária para o modelo em

estrela. Normalmente, e para que seja possível garantir as restrições de integridade de forma implícita, primeiro são carregadas ou atualizadas as dimensões existentes no modelo e só depois se obtêm os valores das chaves primárias dos registos necessários para descrever os factos. De seguida, procede-se ao seu carregamento nas respetivas tabelas de facto. É importante referir que a área temporária não serve unicamente para guardar transitoriamente os valores extraídos dos sistemas fonte. É, igualmente, aqui que podem ser guardados meta-dados referentes a todo este processo de *ETL*.

Apesar da descrição feita das fases do processo de *ETL*, pode-se ficar com a ideia de que este processo é simples de implementar, quando, na verdade, não é. De acordo com Ralph Kimball, pioneiro no desenvolvimento deste tipo de projetos, metade de todo o esforço necessário para criar um projeto como este encontra-se neste módulo. Para além disso, segundo Kimball, é na implementação do processo *ETL* que cerca de 70% dos projetos falha. São números como estes que mostram a importância e a atenção que se deve dar a este processo.

Após a descrição daquilo que o processo de *ETL* deve fazer, a arquitetura deste módulo torna-se simples de compreender. Na figura 9 que se segue, pretende-se mostrar que o processo de *ETL* não é mais que uma sequência de passos que movem dados do sistema fonte para a *data mart*.

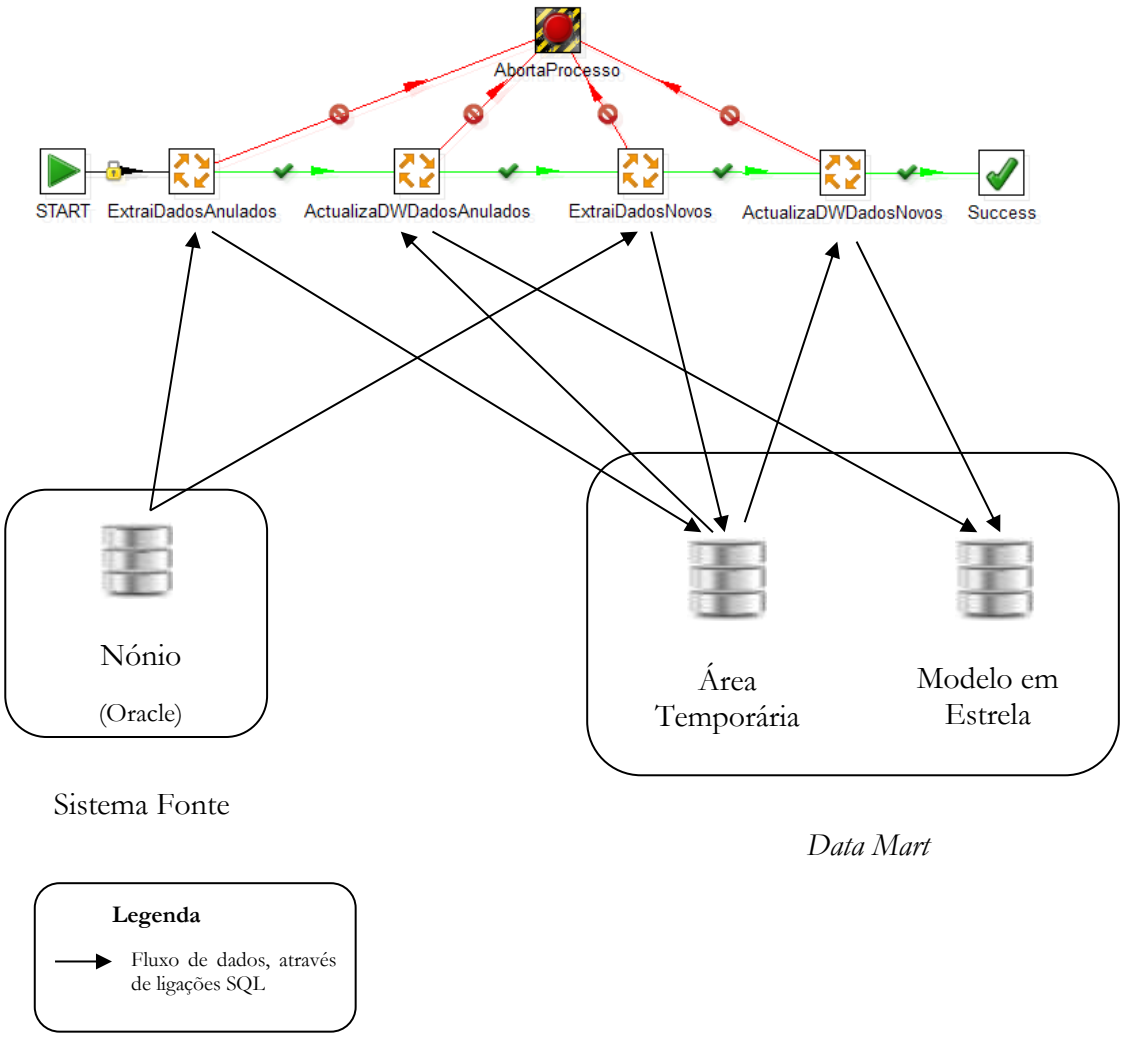

Figura 9 - Arquitetura do processo de *ETL*

A figura 9 representa uma versão simplista da mostrada no capítulo seguinte. Aqui, pretende-se somente expor o fluxo de dados entre o sistema fonte, a *data mart* e os processos *ETL*.

Como podemos observar, existe um conjunto de transformações que formam uma sequência de passos. É através desta sequência que o modelo em estrela é atualizado. Em primeiro lugar, são estabelecidas ligações *SQL* ao Nónio e à *data mart*, mais concretamente à área temporária, para que seja possível extrair a informação. No segundo passo, a informação colocada na área temporária é processada de modo a ser inserida no modelo em estrela. Estes dois passos concluem o processo *ETL* para atualizar o modelo em estrela com os dados anulados no sistema fonte. Os restantes dois passos repetem o mesmo fluxo de dados apresentado, mas, agora, os dados carregados para o modelo em estrela são os dados novos que existam no Nónio. Os passos são executados em série, no entanto, caso um deles falhe, todo o processo é abortado.

4.5. *OLAP*

Este módulo é responsável por fazer a ponte entre os *dashboards* que são apresentados ao utilizador e os dados que estão na *data mart*. Para isso, é configurado um conjunto de cubos correspondente às estrelas. O objetivo dos cubos é disponibilizar uma estrutura onde se podem aplicar as operações de *drill-down*, *roll-up*, *slice*, *dice e drill-across* e executar *queries MDX*.

Para se tornar explícito o modo como o servidor *OLAP* executa as *queries MDX* de modo a obter o resultado da análise, é necessário explicar alguns conceitos.

O primeiro conceito que requer uma explicação é o conceito de esquema. Um esquema consiste num conjunto de cubos e num conjunto de dimensões. É através deste artefacto que o *Mondrian*[11], ferramenta usada no servidor *OLAP* para a definição dos cubos, consegue saber quais as dimensões e as estrelas presentes na *data mart*.

As dimensões são artefactos que mapeiam as dimensões existentes no modelo em estrela nas dimensões dos cubos, definidos no servidor *OLAP*. É nestes artefactos que são especificados os nomes das tabelas dimensão do modelo em estrela e as colunas que possibilitam que nos cubos sejam feitas agregações. Nas dimensões, as colunas podem ser agregadas em hierarquias. Para a explicação do conceito de hierarquia, vejamos o seguinte exemplo. Se tivermos uma dimensão com o tempo em anos civis e em anos letivos, ou seja, na tabela correspondente à dimensão tempo, no modelo em estrela, temos colunas com o trimestre, o semestre e o ano letivo e civil para cada dia do ano. Ao especificar a dimensão tempo no servidor *OLAP*, podemos criar duas hierarquias na dimensão: a hierarquia correspondente ao tempo letivo, que conterá informação sobre a coluna do trimestre, do semestre e do ano letivo, e a hierarquia do ano civil, que conterá informação sobre as colunas do trimestre, do semestre e do ano civil. Deste modo, é possível, usando apenas uma dimensão, a análise sob dois pontos de vista diferentes dos factos presentes nos cubos.

O cubo é um artefacto que define o mapeamento entre uma tabela de factos, as dimensões que lhe estão associadas e os factos existentes na tabela de factos. É uma estrutura que contém o mapeamento entre as chaves forasteiras das tabelas de facto e as chaves primárias das dimensões. Cada facto especificado no cubo corresponde a uma coluna da tabela de factos e a uma operação de agregação, que pode ser uma soma ou uma média, por exemplo.

Após a definição dos cubos e das dimensões, através de um esquema, a arquitetura deste módulo é agora de mais fácil compreensão. Sempre que uma *query* em *MDX* é pedida para ser executada, o que o servidor faz é ir aos cubos e às dimensões especificadas no esquema e

traduzir a *query* em *MDX* para uma *query* em *SQL*, com base nos mapeamentos especificados em cada cubo e dimensão, presentes na *query MDX*.

Para além de executar *queries* em *MDX*, o servidor de *OLAP* tem uma *cache* temporária de resultados, o que permite que, ao executar uma *query MDX,* o resultado devolvido possa provir, total ou parcialmente, da *cache*.

4.6. Interface *Web*

Este módulo é responsável pela apresentação dos *dashboards* num *browser* para que o utilizador possa interagir com eles. Na figura 10, é detalhada a arquitetura do módulo.

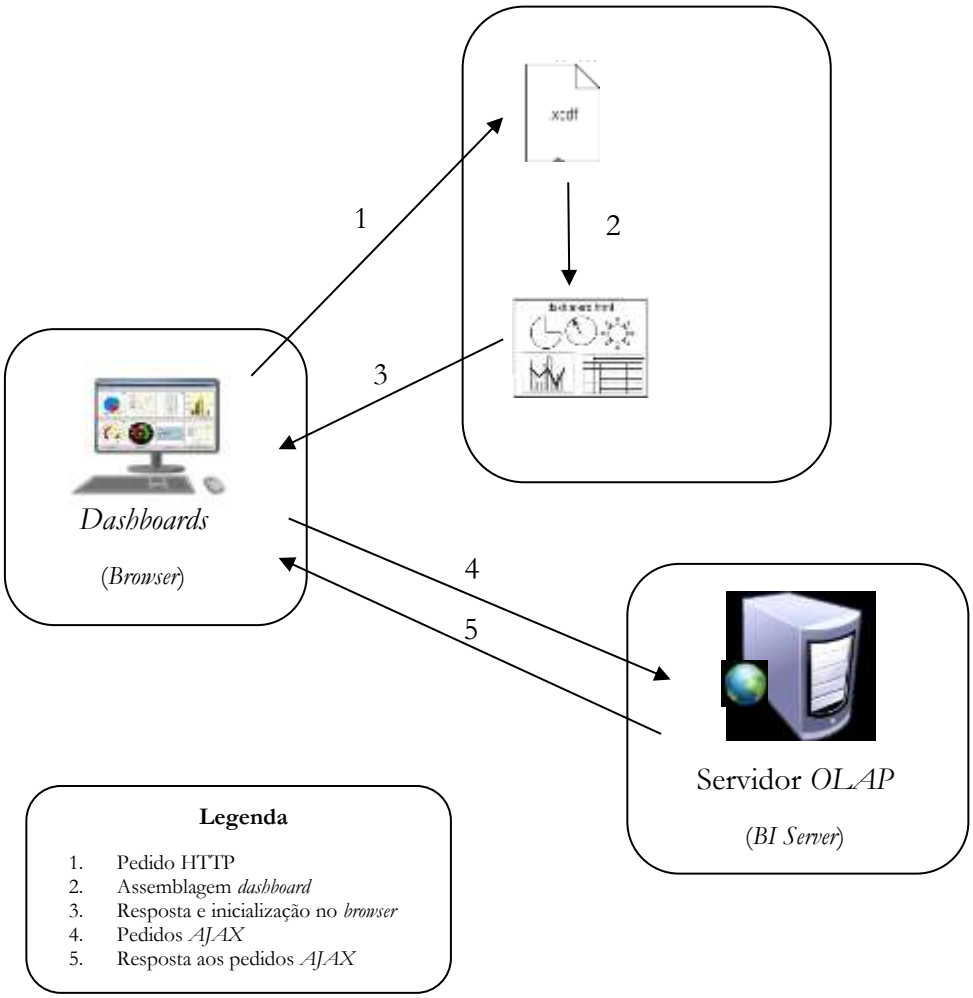

Figura 10 – Inicialização do *dashboard.*

Como é possível observar nesta figura, em primeiro lugar, existe um pedido para um *URL* do *browser* do utilizador. O *plugin* utilizado, ao receber o pedido do utilizador, faz a assemblagem do *dashboard* que o retorna ao *browser* do utilizador. Por sua vez, no lado do utilizador, é feita a inicialização do *dashboard* e, através de pedidos *ajax*, faz-se a inicialização dos componentes do *dashboard*. Sempre que houver uma interação do utilizador com o

dashboard, é feito um pedido *ajax* ao servidor, o qual devolve a resposta ao *browser* do utilizador.

4.7. Seleção de Tecnologia

A tecnologia a usar em cada uma destas fases teve algumas restrições que foram impostas pelo responsável do projeto. A restrição mais importante para a escolha das tecnologias foi que o seu custo fosse nulo. Para além desta restrição, aquando da integração no projeto, havia já um módulo em produção, que pretende mostrar os indicadores definidos para a área dos projetos de investigação. Este módulo tem, como tecnologia de *ETL,* o Pentaho Data Integration, usa, como motor de base de dados, o Oracle e, como servidor *OLAP,* o Pentaho BI Server.

Dado que houve a necessidade de trocar a base de dados, irá usar-se, para este projeto, PostgreSQL. A escolha desta base de dados está, fundamentalmente, ligada ao facto de ser uma base de dados *open-source* e que oferece uma grande quantidade de funcionalidades, quer ao nível do armazenamento de dados, quer ao nível de aceleração de *performance*, através de estruturas como índices ou vistas materializadas. Para além disso, estão presentes, no *website*[12] do motor de base de dados, referências que relatam o excelente desempenho deste motor de base de dados, em ambientes empresariais.

Para os processos de *ETL* e *OLAP,* usou-se o *software* que já tinha sido escolhido para o módulo dos projetos de investigação, dado que o *feedback* obtido foi ótimo. Destaca-se a simplicidade de utilização de ambos os sistemas, isto é, a curva de aprendizagem é pouco inclinada e tem uma excelente *performance*, quando colocados em ambientes com grandes quantidades de dados para processar. Para além destes produtos, será necessário utilizar uma biblioteca específica para o desenho dos gráficos, presentes no protótipo efetuado, aquando da análise de requisitos. A biblioteca a usar será Highcharts[13].

Não obstante estas condicionantes na escolha da tecnologia para cada um dos módulos da arquitetura, foi feita uma análise do *software* que poderia ser utilizado. Esta análise permitiu diagnosticar os pontos fortes e fracos de cada tecnologia e as características chave a que a tecnologia selecionada deveria ser capaz de dar resposta para cada módulo da arquitetura. Este estudo encontra-se no anexo 1.

Capítulo 5
Implementação

Neste capítulo, é apresentado o modo como as várias componentes da arquitetura foram implementadas e os pormenores de implementação, sempre que necessário. São, aqui, também apresentadas as maiores dificuldades encontradas, durante o desenvolvimento das várias fases da arquitetura e as soluções encontradas para as superar.

5.1. *Data mart*

Tal como foi referido no capítulo anterior, para simplificar o desenvolvimento do processo de *ETL*, foram criados dois modelos de dados na *data mart*: a área temporária e o modelo em estrela.

Detalhes da implementação da área temporária

A área temporária, apresentada no capítulo anterior, não é mais do que um conjunto de tabelas avulsas numa base de dados. Decidiu-se implementar a área temporária deste modo, porque o objetivo principal era tornar o desenvolvimento do modelo de dados e dos processos de extração dos dados e carregamento da área temporária mais simples e rápidos. Para além disso, dado que não existem relações de integridade entre as tabelas, isto permitiu que os processos de *ETL* pudessem ser executados em paralelo, conseguindo, desta forma, aumentar a *performance* e diminuir o tempo de interação com o sistema do Nónio.

Considerando que os dados podem mudar ao longo do tempo e que nem sempre podem estar completos no sistema do Nónio, optou-se por não colocar quaisquer restrições nas colunas das tabelas da área temporária, com o objetivo de maximizar o número de registos passíveis de serem extraídos do sistema e, ao mesmo tempo, evitar que o processo de extração dê erro ao guardar os registos, nas tabelas da área temporária.

Como não existem restrições nas colunas das tabelas da área temporária, isto significa que não existem colunas com a designação de chave primária ou chave forasteira e, consequentemente, não existem índices sobre essas colunas. Para se obter alguma *performance* nas *queries* que são executadas nas tabelas da área temporária, decidimos, inicialmente, adicionar índices nas colunas que, em condições normais, teriam este tipo de restrição.

Após o desenvolvimento dos processos de *ETL,* estando já na fase de otimização do tempo de execução do processo de *ETL*, investigámos quais as colunas das tabelas, presentes na área temporária, em que seria mais vantajoso criar índices. De acordo com o livro *Database System Concepts*[14], concluímos que as colunas que fazem parte das condições nas cláusulas *WHERE* seriam boas candidatas. Criámos, ainda, índices em colunas, cujo resultado devolvido seria, à partida, um conjunto muito pequeno, em relação ao tamanho total da tabela.

Após a adição dos índices, o tempo de execução das *queries* encontra-se agora em valores de cerca de três a quatro segundos. É preciso notar que estes valores de execução compreendem a junção entre cinco e seis tabelas com várias restrições, podendo também haver operações de agregação de dados. Das tabelas consideradas na junção, a maior tem mais de trezentos mil registos e as mais pequenas têm cerca de duzentos.

Apesar destas otimizações, o tempo de carregamento das tabelas ainda estava longe de ser o ideal e, por isso, optou-se por investigar uma maneira de evitar que as transações fossem

escritas para o *log* da base de dados. À primeira vista, esta solução pode parecer menos correta, uma vez que se perde o fator durabilidade, no caso de a base de dados terminar a execução abruptamente ou ocorrer outro evento que impeça o normal funcionamento do sistema, mas, como o processo de *ETL* foi implementado com sendo uma transação com vários pontos de controlo, esta solução faz todo o sentido. Assim, caso a base de dados falhe e seja necessário reiniciar o processo de *ETL*, apenas as secções a seguir ao último ponto de controlo executado com sucesso serão executadas. Após a investigação, verificou-se que, alterando ligeiramente parte da sintaxe de criação das tabelas na base de dados de *CREATE TABLE* para *CREATE UNLOGGED TABLE*, esta alteração faz o que se pretende. Ao adoptar esta abordagem, não houve a necessidade de eliminar os índices sobre as colunas antes dos processos de inserção, dado que o processo responsável pela inserção de dados novos demora cerca de três minutos a introduzir cerca de trezentos mil registos na maior tabela.

Todas as tabelas, à exceção da tabela metadata, foram alteradas para esta nova sintaxe, pois é com base nesta tabela que o processo de *ETL* controla os pontos de controlo que já executou com sucesso. Portanto, para que o processo de *ETL* consiga perceber quais os novos dados, anulados e não anulados, são guardadas, na tabela metadata, as datas mais recentes das referências e das parcelas de compensação, que foram anuladas, em relação à data atual. Salienta-se ainda que o maior valor das chaves primárias das referências e das parcelas de compensação também será mantido. Deste modo, o processo de *ETL* saberá sempre quais foram os últimos registos processados e, na execução seguinte, processará simplesmente os que foram alterados ou criados, entre a data da última execução e a data da execução atual, tendo em conta os maiores valores da chave primária para os novos registos e o intervalo entre a data da última referência anulada, registada na tabela metadata, e a atual.

Detalhes da implementação do modelo em estrela

Com a adição de três novos indicadores, taxa de incobrabilidade, propina de cobrança duvidosa e discriminação da receita mensal por anos letivos, houve a necessidade de acrescentar mais tabelas de facto para acomodar as novas análises. Foram adicionadas duas tabelas, para mensurar a evolução da receita discriminada por anos letivos, uma somente para as propinas e outra para os juros. Foi, ainda, adicionada mais uma tabela para a dívida de propinas que se encontra na categoria de cobrança duvidosa. Para a taxa de incobrabilidade, não foi necessário criar nenhuma tabela adicional, pois trata-se de operações aritméticas entre a dívida e a receita de propina.

A segunda alteração ao modelo de dados ocorreu no momento em que percebemos que as tabelas de facto seriam insuficientes para mensurar os indicadores quando havia situações especiais envolvidas. Uma vez que uma matrícula de um aluno pode ter várias situações especiais, atribuídas num mesmo ano letivo, ocorre uma violação da restrição de N:1 entre as tabelas de facto e as dimensões apresentadas no modelo em estrela. A relação passa a ser de N:M. Para resolver este desafio, foi criada uma tabela de factos para todos os indicadores, em que é possível fazer uma análise por situação especial com granularidade diferente. Deste modo, alguns indicadores têm duas tabelas de facto, uma com granularidade de referência ou parcela de compensação e outra com granularidade de referência por situação especial ou parcela de compensação por situação especial. Isto significa que, nas tabelas de facto, as referências ou parcelas de compensação, que contenham mais que uma situação especial por ano letivo, irão aparecer várias vezes, uma por situação especial, permitindo, deste modo, discriminar os indicadores por situação especial.

Visto que não houve mais nenhuma alteração ao modelo em estrela e, após o desenvolvimento dos *dashboards,* constatámos que as *queries* executadas para apresentar os

resultados estavam a demorar tempo considerável a serem processadas, então, decidiu-se aplicar o mesmo raciocínio, utilizado na área temporária, para criar índices nas tabelas. Naturalmente, foram adicionados índices nas colunas das tabelas de facto que contêm as chaves forasteiras para as dimensões que as descrevem e índices nas colunas das dimensões que mais comumente aparecem nas cláusulas *WHERE*.

Dado que, em alguns casos, a adição de índices não foi suficiente para aumentar o desempenho, a título experimental, foi adicionada uma tabela de agregados, relativamente à tabela de factos responsável pelo indicador dos meios de pagamento remotos. Para criar o agregado, foram tidas em consideração três das quatro dimensões presentes na tabela de factos original. Desta forma, foi possível reduzir o número de registos em relação ao da tabela original para cerca de 4%. Na prática, passou-se de uma tabela de factos, com cerca de cento e oitenta e três mil registos, para uma tabela de factos que tem cerca de sete mil. Para popular a tabela, bastou inserir os registos provenientes de um *SELECT* que faz a soma dos factos agrupados pelas dimensões que foram selecionadas na tabela de agregados.

Após esta otimização, o desempenho da visualização dos dados nos *dashboards,* relativos aos meios de pagamento, aumentou para valores aceitáveis. Como esta experiência foi realizada com sucesso, esta estratégia será aplicada a todas as tabelas de facto. Dada a dimensão do modelo de dados e o facto de que o número de agregados, que pode ser implementado em cada tabela de factos, resulta da combinação das dimensões nela presentes, foram apenas criados agregados com base em *queries* mais comuns, presentes nos *logs* do *Mondrian* no servidor de produção. Quando for necessário, mais agregados poderão rapidamente ser adicionados.

5.2. Extração, Transformação e Carregamento

O processo de *ETL*, apresentado no capítulo anterior, é composto por duas partes distintas. A primeira parte refere-se à extração e carregamento dos dados que foram anulados no sistema do Nónio, relativamente à cobrança de propinas e dados académicos de alunos, e a segunda parte tem a ver com a extração e carregamento dos novos dados.

Para evitar repetir uma das partes, no caso de o processo de *ETL* abortar, durante a sua execução, existem duas transformações que atualizam a tabela metadata, apresentada na secção anterior. Assim sendo, caso o processo de *ETL* aborte a sua execução, após o ponto de controlo relativo ao carregamento das referências e compensações anuladas ou caso não existam valores novos, na próxima execução, esta parte do processo não será executada.

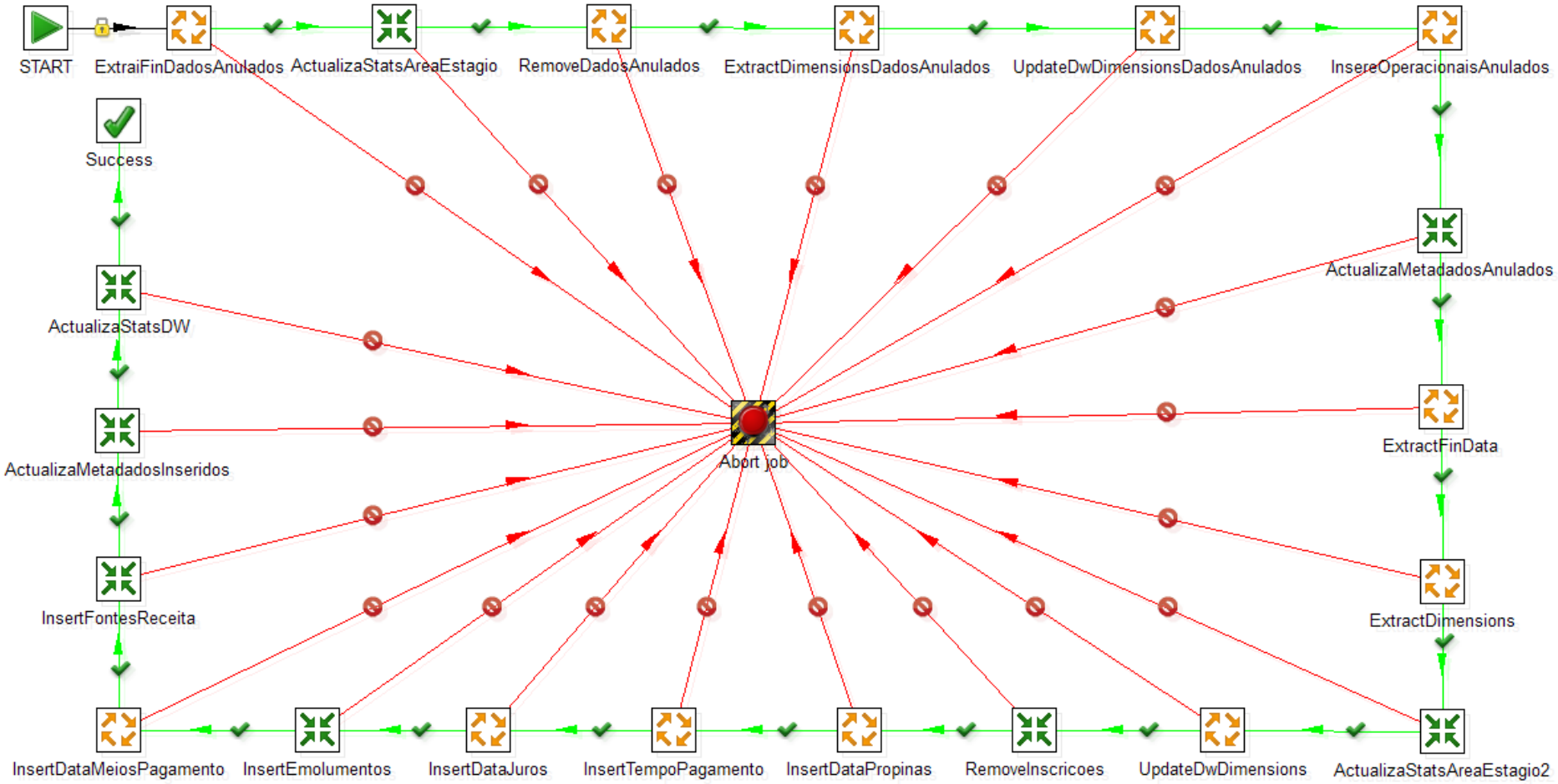

Figura 11 - Processo de *ETL*, responsável por atualizar todo o modelo em estrela.

Como se pode observar na figura 11, numa primeira transformação (ExtraiFinDadosAnulados) são carregados os dados anulados do Nónio para a área temporária. Após as tabelas recebimento e prestação terem sido preenchidas, as restantes serão carregadas na transformação seguinte com base nos *id's* de matrícula e curso presentes em ambas as tabelas, inicialmente carregadas. Após o carregamento destas tabelas com a informação dos cursos, UOs, situações especiais, regime e demografia, o processo que se segue está encarregado de atualizar as dimensões do modelo em estrela (UpdateDWDimensionsDadosAnulados). As dimensões têm de ser atualizadas antes das tabelas de facto, porque as colunas das tabelas de facto têm relações de integridade que verificam a existência dos *id's* nas respetivas dimensões e, caso não existam, a inserção dos factos não iria ser possível. Após a atualização das dimensões, tratámos, então, de inserir factos anulados, no caso dos indicadores que medem o desempenho da receita de propinas e juros em relação ao ano civil, e removemos os dados anulados, nos restantes indicadores (InsereOperacionaisAnulados).

Optou-se por remover os dados anulados para poder tirar partido dos índices existentes nas tabelas de facto. Dado que todas as tabelas de facto têm índices nas colunas onde se encontram guardados os i*d's* das referências e das parcelas de compensação, todo este processo é executado de forma mais rápida.

Seguidamente o processo responsável por carregar os dados novos existentes no Nónio faz a atualização do modelo de dados, de forma bastante similar ao descrito anteriormente. Assim sendo, o carregamento da área temporária é realizado do mesmo modo, com a exceção de que, agora, são trazidos apenas os novos dados (ExtractFinData). Para atualizar as tabelas de facto, é feita a inserção dos registos sempre que estes não existam.

Na figura 12, apresenta-se um processo de *ETL* que faz a atualização de uma tabela de factos, relativa ao indicador da propina devida.

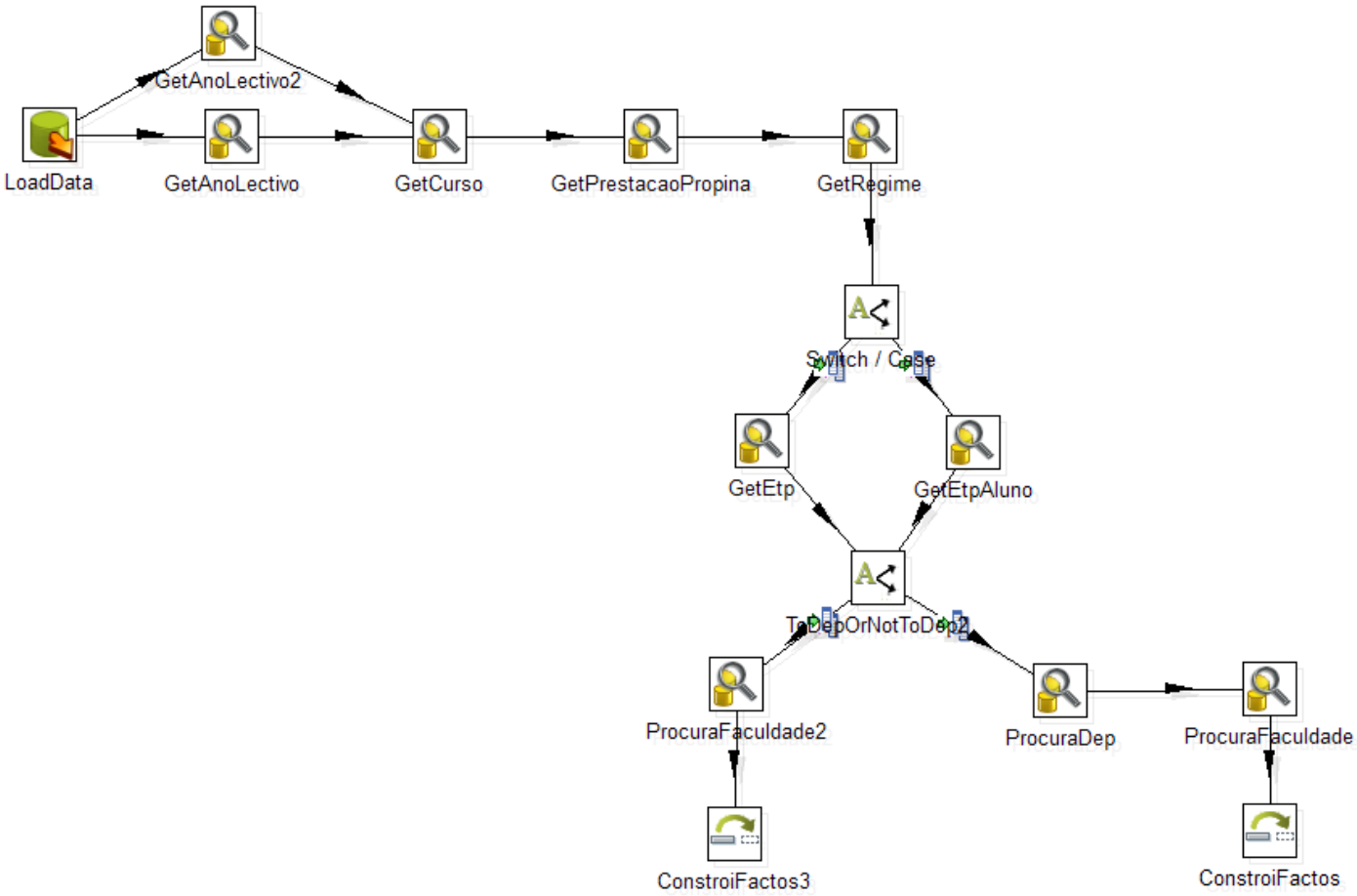

Figura 12 - Exemplo de um processo de *ETL* que atualiza uma tabela de factos

Como é visível na figura 12, o processo de *ETL* começa por executar uma *query* na área temporária no passo LoadData. Esta *query* faz a junção da tabela prestação com as tabelas do regime, curso e unidade orgânica, cujo resultado é um registo de uma referência associado a um ano letivo, curso, unidade orgânica, prestação de propina, regime e entidade pagadora. Após a devolução dos resultados pela *query*, estes são encaminhados para os passos que fazem a procura do *id* do ano letivo na dimensão. Ao colocarmos duas transformações que fazem a pesquisa dos parâmetros do tempo, metade do resultado da *query* é processado em cada uma delas. Foi tomada a decisão de colocar duas transformações a fazer a pesquisa do *id* do tempo, porque se verificou, durante a implementação, que esta transformação era o *bottleneck* de todo o processo. Com a adição desta transformação, este problema foi mitigado.

As transformações Switch/Case e ToDepOrNotToDep2 são responsáveis por redirecionarem o fluxo de dados, consoante a entidade pagadora ou o departamento sejam nulos, respetivamente. Em ambos os casos, é necessário que a chave forasteira a colocar na tabela de factos seja nula. Por essa razão, existem duas transformações, uma que faz a pesquisa na dimensão e outra que retorna sempre o valor *NULL*.

As transformações de nome ConstroiFactos e ConstroiFactos3 são responsáveis por verificar se o registo a inserir já existe e, caso exista, não é inserido. Todos os processos de atualização das tabelas de facto são similares ao apresentado. As únicas diferenças encontram-se na quantidade de transformações, que fazem a pesquisa de *id's* nas dimensões, e na *query* na transformação LoadData. As restantes transformações podem ser encontradas no anexo 1.

Um dos maiores desafios, que foi transversal a todos os processos de *ETL* desenvolvidos, foi ultrapassar as limitações impostas pelo *software*. Como se trata de um *software* com um conjunto pré-definido de transformações que se podem aplicar aos dados, houve uma necessidade constante de alterar alguns processos de *ETL*, porque, durante a sua implementação, ou não havia uma transformação, que era necessária para realizar um passo, ou não era possível contornar a limitação, através de *SQL*, *JavaScript* ou *Java*, as três maneiras disponibilizadas pelo *software* para implementar código genérico.

Ultrapassada esta limitação e especificados todos os processos de *ETL* necessários, foi feita uma medição do tempo do processo de *ETL*, representado na figura 11. Verificou-se que o processo demora cerca de seis horas a executar, concentrando-se na maior parte do tempo nos passos que fazem o carregamento de todos os indicadores das propinas, meios de pagamento e juros. Para otimizar o carregamento, antes do passo incumbido de carregar os dados relativos aos indicadores da propina, todos os índices das tabelas de facto serão eliminados e, após a inserção dos dados relativos aos indicadores dos juros, os índices serão criados de novo. Com esta otimização, o tempo de carregamento desce. Os tempos aqui apresentados dependem, essencialmente, da quantidade de dados que houver para atualizar. Como estes valores são relativos a primeiros carregamentos, isto é, a primeira vez que o processo de *ETL* corre para fazer o carregamento inicial das tabelas de facto, espera-se que carregamentos sucessivos, como terão muito menor quantidade de dados, sejam executados em menos tempo.

Por fim, para verificar que o processo de *ETL*, apresentado na figura 11, executava periodicamente no tempo, foi agendada a sua execução no *scheduler* do sistema operativo. Após consulta dos *logs* do processo de *ETL*, concluimos que este se encontrava a executar nos dias e horas para os quais tinha sido definido. Apenas foi agendado este processo, porque, como já foi referido, serve para atualizar todos os indicadores modelados neste módulo.

5.3. Cubos

O *Mondrian*, ferramenta que faz a gestão dos cubos e da *cache* de resultados no servidor *OLAP*, necessita de conhecer os cubos que são criados, a partir do modelo em estrela, presente na *data mart*. Para isso, utilizou-se um *software* chamado *schema workbench* da Pentaho, criado com o objetivo de tornar a especificação do *XML* do esquema, dimensões e cubos mais intuitiva.

O primeiro passo foi criar um novo esquema que continha todas as dimensões, cubos e cubos virtuais que foram necessários para a análise deste projeto. Após a criação do esquema, adicionámos as dimensões.

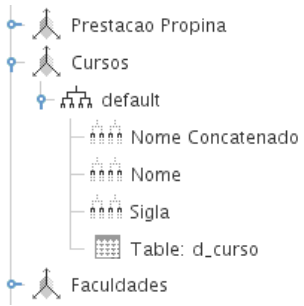

Figura 13 - Exemplo de especificação de dimensão no cubo

Para a adição da dimensão cursos, representada na figura13, em primeiro lugar, foi necessário atribuir-lhe um nome e a tabela correspondente no modelo em estrela, neste caso, a tabela d_curso. Depois, bastou adicionar níveis, ou seja, mapear colunas. Nesta figura, temos o exemplo das colunas Nome Concatenado, Nome e Sigla como sendo níveis pelos quais o *Mondrian* poderá realizar operações de *slice*. Tendo em conta as dimensões pelas quais será feito *slice,* nos diversos indicadores, no decorrer da implementação deste projeto, foram especificadas dezoito dimensões. Após a especificação das dimensões, passámos à especificação dos cubos.

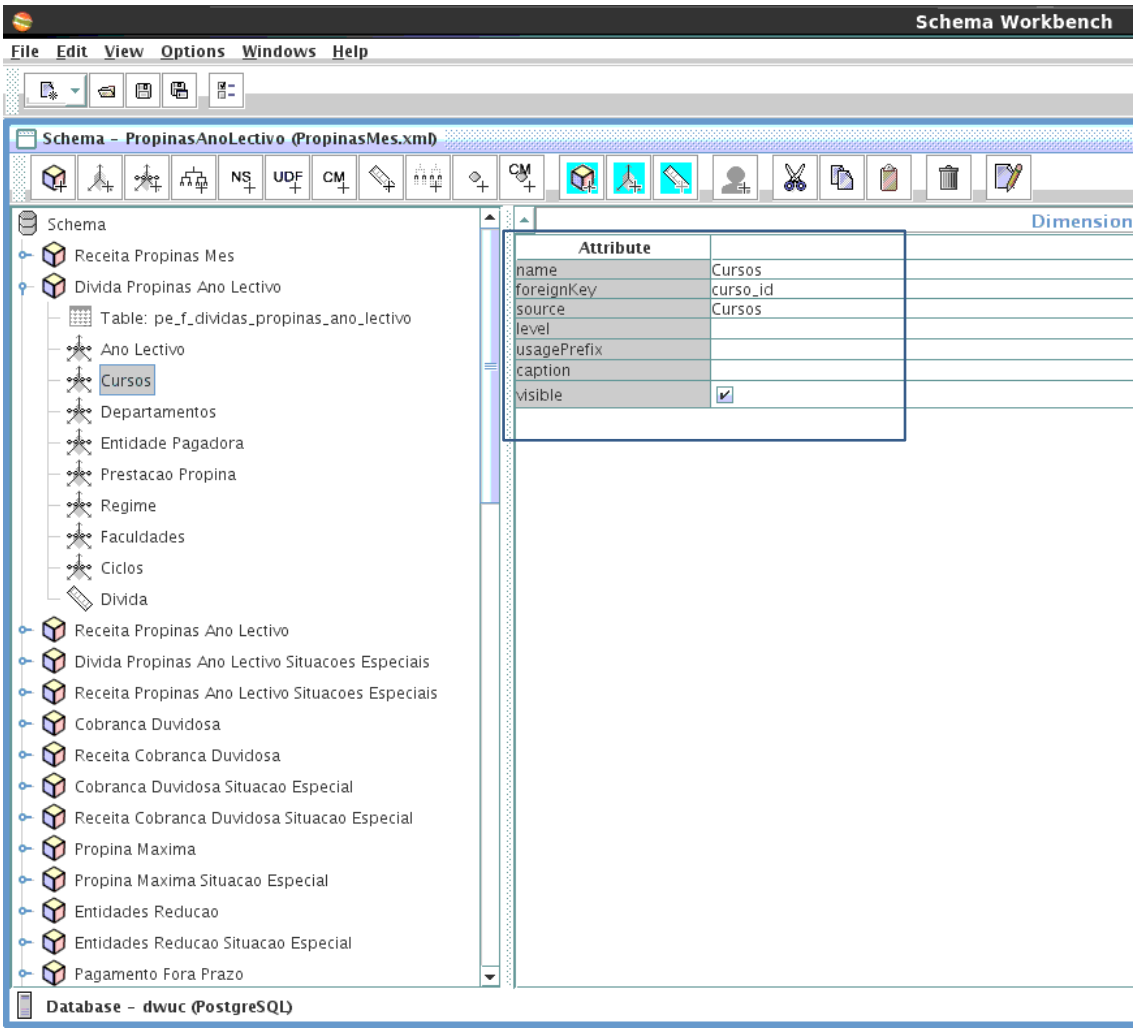

Figura 14 - Exemplo de especificação de uma tabela de factos.

Na figura 14, encontra-se o exemplo de como um cubo deve ser especificado. Existem algumas semelhanças com o método usado nas dimensões. Desde logo, foi necessário atribuir um nome e mapear uma tabela do modelo em estrela. Neste caso, a tabela a ser mapeada foi a tabela de factos pe_f_dividas_propinas_ano_letivo. Seguidamente, foram referenciadas as chaves forasteiras da tabela de factos nas dimensões presentes no cubo com o nome dado às dimensões anteriormente criadas. Tomando o cubo da figura, como exemplo, para especificar a dimensão Curso, podemos afirmar que a chave forasteira é curso_id, que corresponde ao nome da coluna, na tabela de factos, e que a dimensão a usar é a dimensão Curso, especificada anteriormente.

Por fim, foi necessário acrescentar os factos e a função de agregação. Para isso, bastou seguir o exemplo ilustrado na figura 15. Só foi necessário dar um nome ao facto, especificar a função de agregação, que, neste caso, é *sum* e especificar a coluna da tabela de factos que está encarregada de guardar os factos.

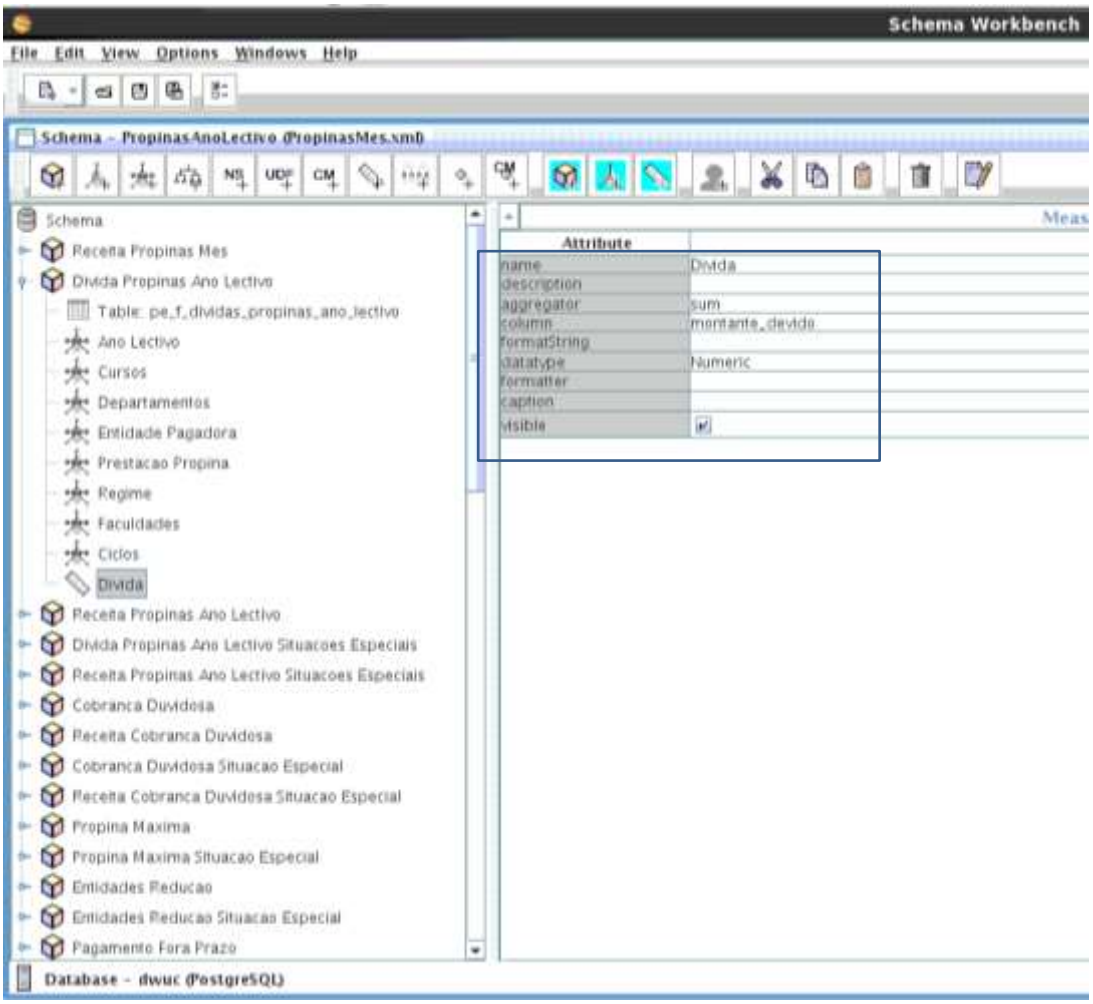

Figura 15 - Exemplo de especificação de um facto no cubo

No decorrer de todo este projeto, especificaram-se vinte e oito cubos, correspondentes às várias tabelas de facto com as diversas granularidades de cada indicador.

Após a implementação dos cubos, houve um grande desafio a superar: a definição dos cubos virtuais, estruturas que permitem realizar a operação de *drill-across*. Dado que não existe muita documentação sobre a sua definição, todo o processo de definição dos cubos virtuais foi realizado recorrendo ao método tentativa/erro. Deste modo, cada vez que se fazia *deploy* do esquema para o servidor de *OLAP*, tinha-se de consultar os *logs* para localizar o erro e para o tentar resolver, com base na mensagem apresentada. Esta tarefa nem sempre foi fácil, dado o conteúdo geralmente incompreensível das mensagens. Após a superação deste desafio implementaram-se nove cubos virtuais, para que, ao fazer *slice* de alguns indicadores, fosse possível realizar *drill-across*. Os indicadores que, regra geral, beneficiaram de um cubo virtual foram os indicadores onde existem tabelas de facto com a granularidade referência por situação especial ou parcela de compensação por situação especial, que necessitam de ser analisadas juntamente com as tabelas de facto, cuja granularidade é referência ou parcela de compensação, respetivamente.

Na figura 16, é apresentado um exemplo de um cubo virtual. Para a sua definição, bastou selecionar as dimensões pelas quais se pretendeu fazer *slice*, tal como nos cubos apresentados anteriormente, e selecionar os factos pelos quais queremos fazer *drill-across*. Ao selecionar os factos, foi necessário especificar de que cubo é que provinham.

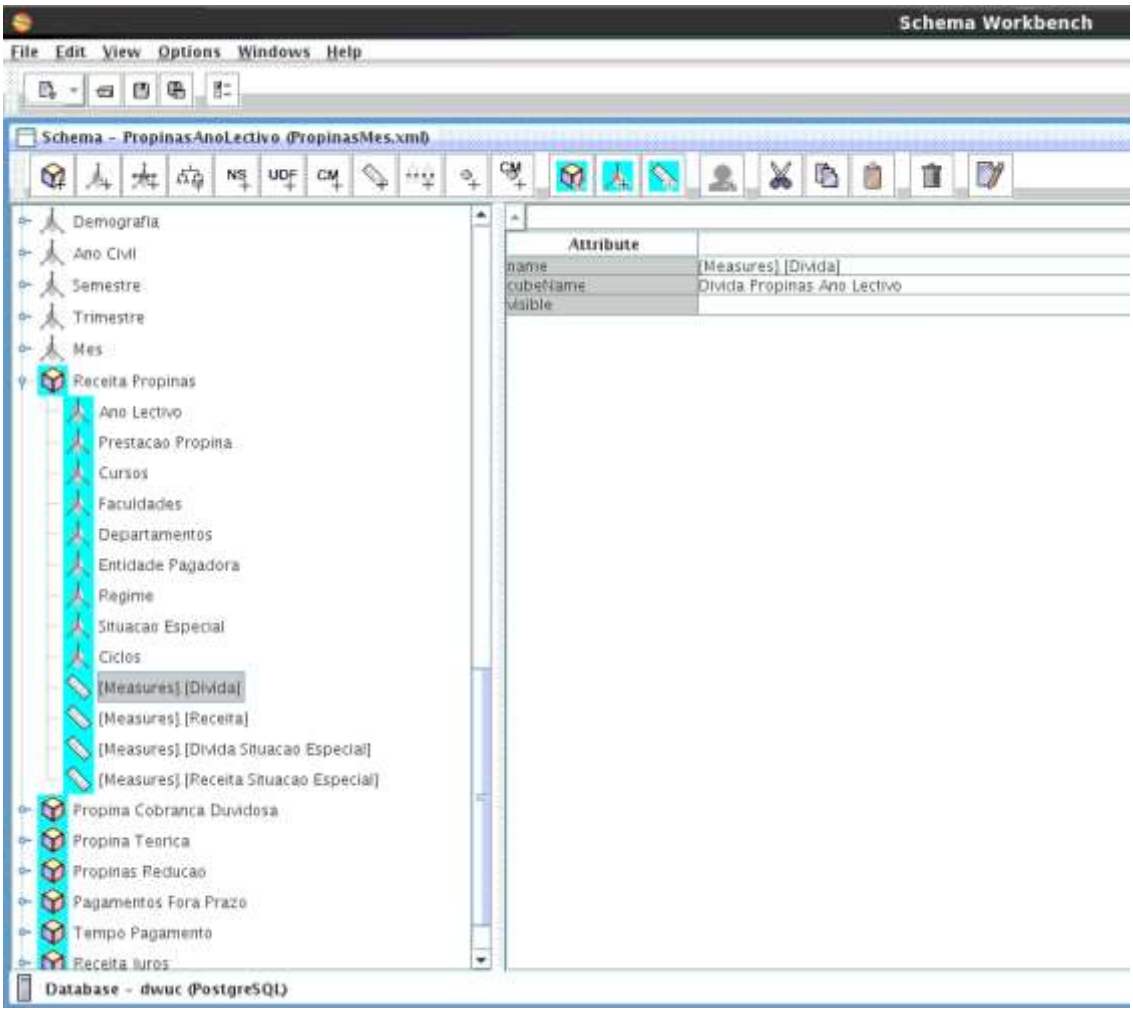

Figura 16 - Exemplo de especificação de um cubo virtual

Após a definição dos cubos e de termos os *dashboards* já quase totalmente implementados, surgiu o desafio de sabermos como especificar nos cubos os agregados que, anteriormente, tinham sido criados em tabelas, no modelo em estrela. Após alguma investigação, concluímos que bastava especificar o agregado na tabela do cubo, tal como se pode ver na figura 17.

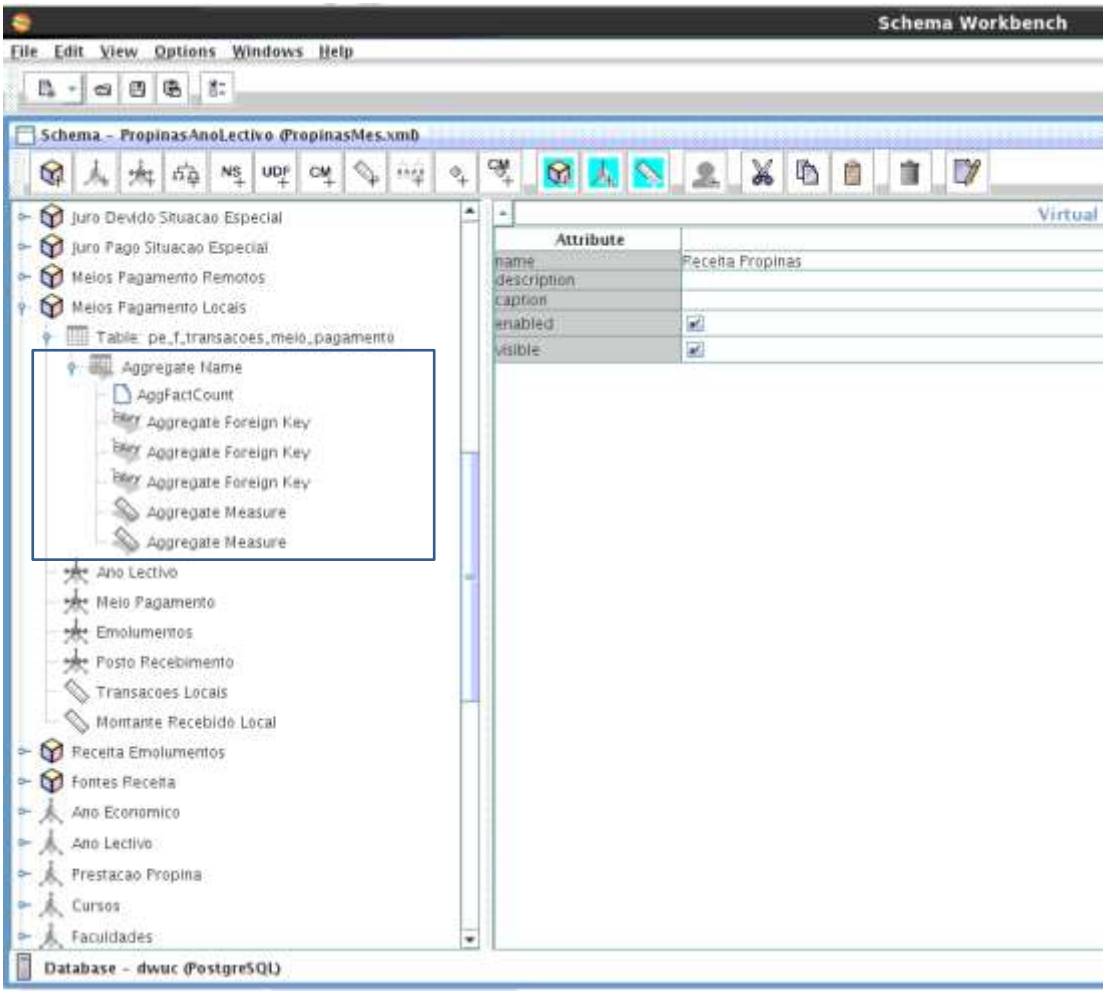

Figura 17 – Exemplo de especificação de agregados no cubo

Ao observar a figura 17, com atenção, para além da especificação das chaves forasteiras e do facto, existe uma coluna chamada *agregate fact count*. Esta coluna é obrigatória na especificação dos agregados, pois é através dela que o *Mondrian* decide se numa determinada *query* usa a tabela com os valores dos agregados ou a tabela de factos que dá origem ao cubo. Atualmente, o *Mondrian* está a decidir qual tabela usar, com base apenas no número de registos, no entanto pode selecionar a tabela, com base no tamanho dos registos em *bytes*, isto é, o *Mondrian* computa uma estimativa do tamanho dos registos de cada uma das tabelas e usa aquela que apresentar menor valor.

O terceiro desafio que foi levantado, durante a definição dos cubos, prendeu-se com o problema de saber como fazer o *Mondrian* calcular o desvio padrão de uma amostra de registos, resultante de uma *query MDX*, como se pode ver na figura 18. Para resolver este problema, começou-se por conceber um novo tipo de dados, contendo apenas um atributo do tipo *double*. De seguida, foi criada uma função *count* que recebe determinado tipo de dados e devolve o seu valor. Dado que as funções de agregação escolhidas na definição dos factos têm correspondência com as funções de agregação existentes na base de dados, na definição deste atributo, especificou-se, como função de agregação, a função *count*.

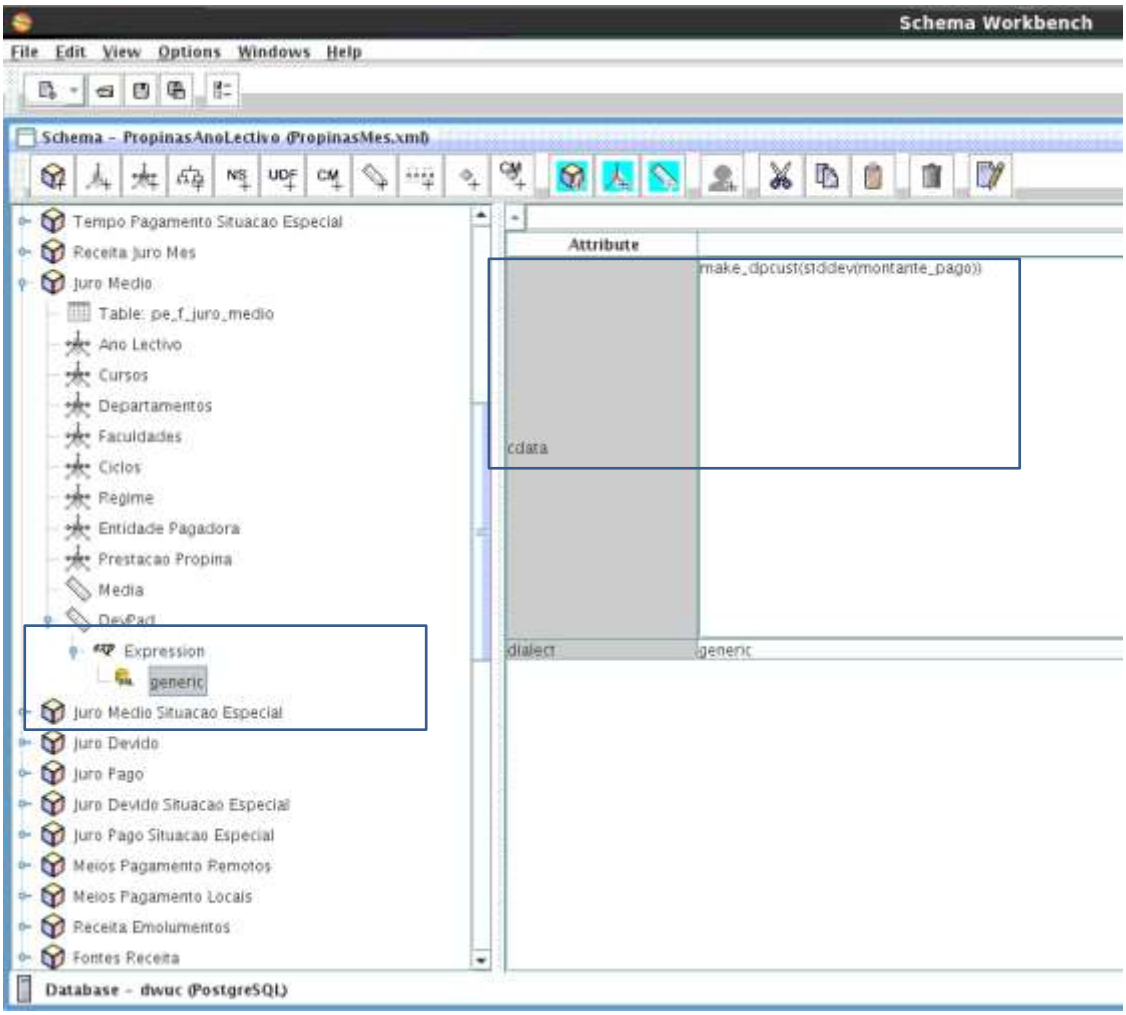

Figura 18 - Especificação do desvio padrão num cubo

5.4. *Dashboards*

Os *dashboards* são os componentes em todo este projeto com os quais o utilizador interage. É também o local onde são apresentados os resultados de todos os indicadores definidos para este projeto e, por isso, é a parte que requer uma maior atenção ao detalhe.

Para implementar um *dashboard*, é necessário saber que este se encontra dividido em três secções. A primeira secção corresponde à secção do *layout*. É aqui que é construído o esqueleto da página *web* onde o *dashboard* irá aparecer. Tudo o que é *html*, *css* e *js*, relacionado com a camada de apresentação, deve aqui ser colocado. A figura 19 pretende mostrar a camada de apresentação de um dos *dashboards* desenvolvidos.

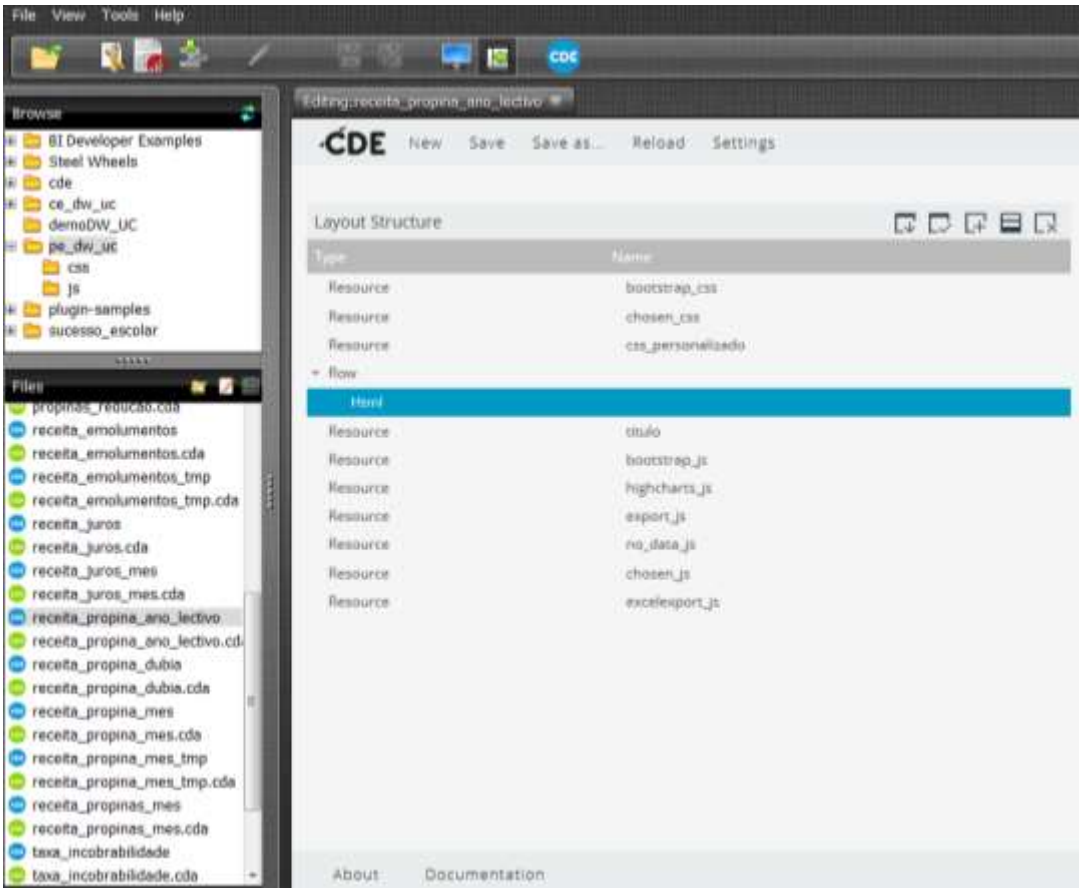

Figura 19 - Especificação do *layout* do *dashboard*

Dentro do campo selecionado, encontra-se todo o *html* necessário para apresentar a vista do *dashboard* ao utilizador. Os restantes elementos do tipo *Resource* são ficheiros no servidor como folhas de estilo ou funções de *JavaScript* que são importados para o *dashboard*.

A segunda secção corresponde às fontes de dados. É aqui que todas as fontes de dados necessárias para popular os gráficos e demais componentes têm de ser especificadas.

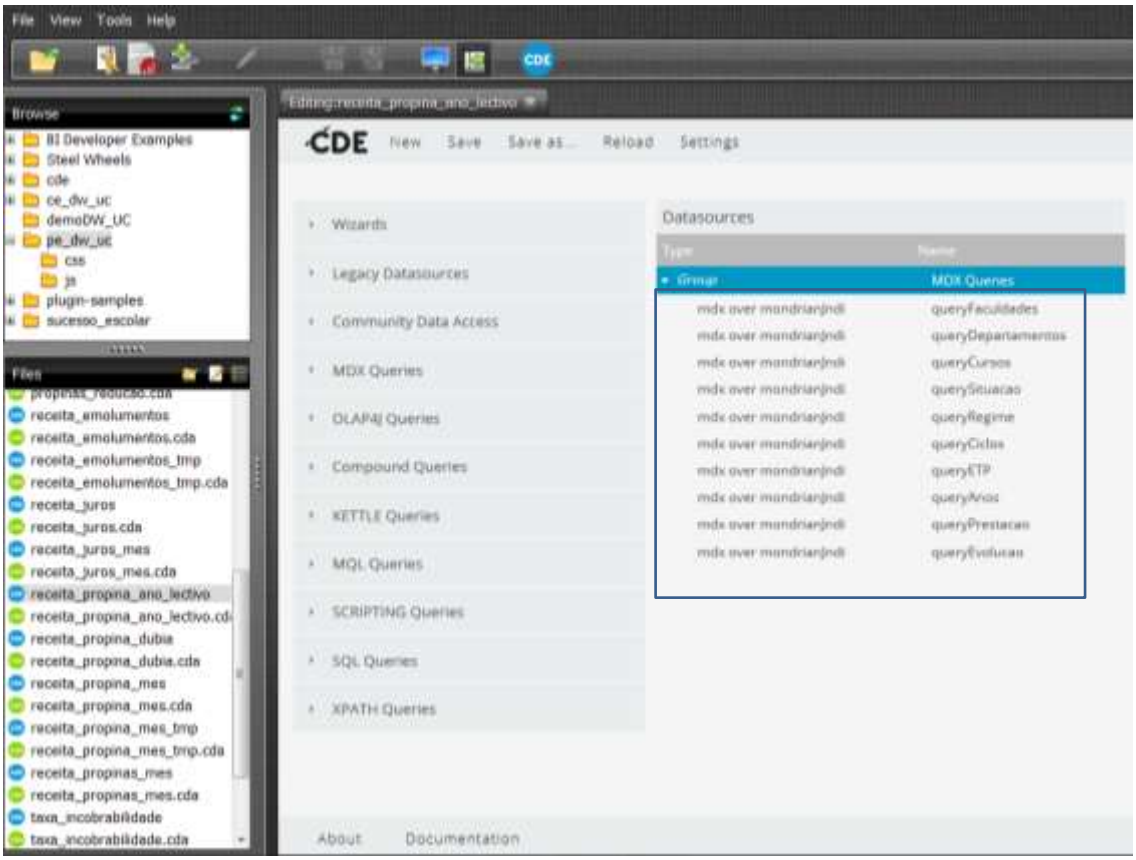

Figura 20 - Exemplo de especificação das fontes de dados num *dashboard*

Na figura 20, podemos observar que foi especificado um conjunto de fontes de dados para preencher os vários componentes do *dashboard*. Existem fontes de dados para as caixas das dimensões, para os gráficos e para as tabelas.

A secção mais importante, na qual foi despendida a maior parte do tempo, é a secção dos componentes. Nesta secção, foram especificados os vários tipos de componentes com os quais o utilizador irá interagir, destacando-se caixas de seleção simples e múltipla, gráficos e tabelas. É aqui que é especificada a fonte de dados para cada componente e a sua localização na página *web*.

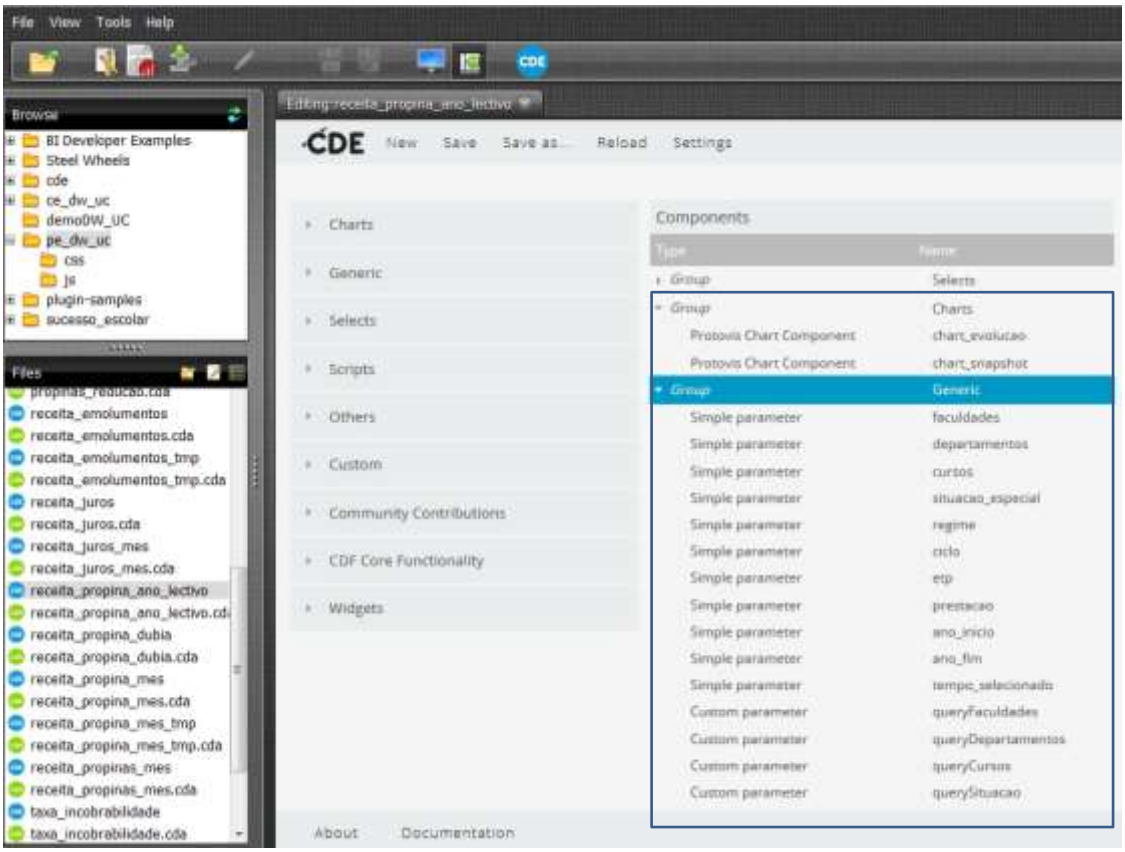

Figura 21 - Exemplo de especificação dos componentes num *dashboard*

O maior desafio que enfrentámos, no desenvolvimento dos *dashboards,* foi a especificação das *queries* necessárias para apresentar os resultados dos *slice* e *dice* nos gráficos, como se pode constatar na figura 21. Visto que as *queries* necessitam de ser computadas de maneira diferente, cada vez que o utilizador seleciona um parâmetro para fazer *slice*, estas são executadas em *JavaScript* dentro de um componente que, depois, alimenta a fonte de dados anteriormente apresentada. Assim, cada vez que o utilizador seleciona um parâmetro pelo qual quer fazer *slice*, existe um componente que é chamado a executar. Nesse componente, há uma função que foi programada em *JavaScript*, com cerca de cinco mil linhas de código, que constrói a *query MDX,* em tempo real.

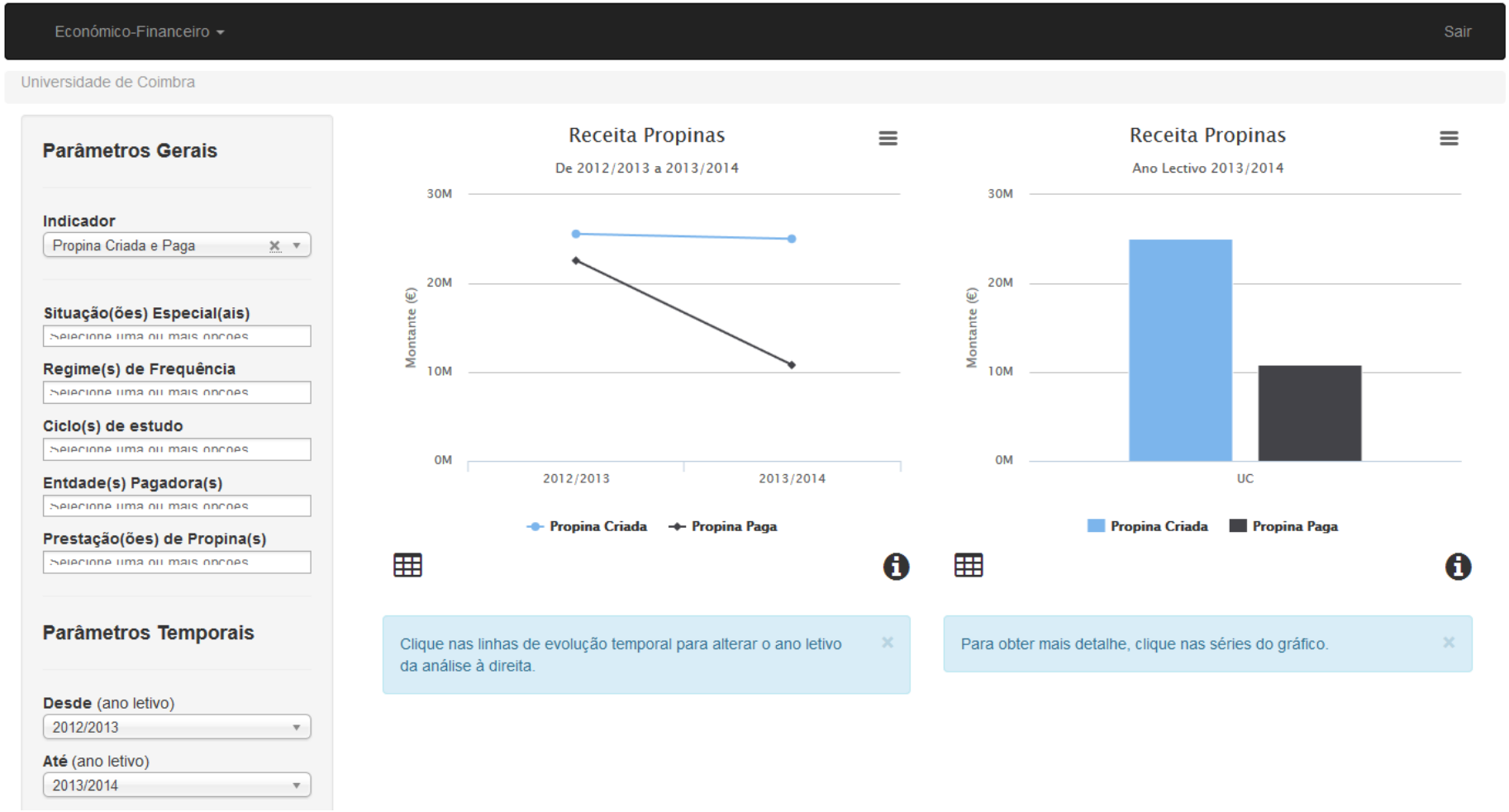

Figura 22 - Exemplo de um *dashboard* implementado

Na figura 22, mostra-se o resultado final de um dos muitos *dashboards* implementados neste projeto de estágio. Para que a dimensão do trabalho desenvolvida seja percetível, é necessário dizer que cada caixa de seleção múltipla, para apresentar valores que podem ser selecionados, tem uma *query* cada uma a ser computada em tempo real que filtra os resultados, com base no que já foi escolhido noutras dimensões. Por exemplo, ao selecionar o campo licenciatura, no ciclo de estudos, e supondo que, na tabela de factos, a interseção do ciclo escolhido com a primeira prestação de propina é um conjunto vazio, a opção da primeira prestação de propina não aparecerá na caixa de seleção da prestação de propina, enquanto licenciatura estiver selecionada, como ciclo de estudos. Tal como no exemplo explicado para a *query* dos gráficos, também aqui cada componente tem cerca de três mil linhas de código em *JavaScript*, para computar a *query*, em tempo real.

No total, este *dashboard* tem cerca de vinte mil linhas de código, sendo a sua maioria funções que constroem as *queries MDX* para mostrar as opções das caixas de seleção e o resultado da análise nos gráficos.

Outro desafio não menos importante que tivemos de resolver foi a funcionalidade de exportar dados da tabela. Apesar de, no editor dos *dashboards,* estarem disponíveis componentes que permitem a exportação de dados para *Excel*, devido ao facto de efetuarmos alterações aos dados, depois da execução da *query* que lhes dá origem e antes de serem apresentados na tabela, ao exportar os resultados, utilizando o componente existente no *dashboard*, estes não tinham a mesma formatação e as alterações efetuadas que eram visíveis nas tabelas. Assim, foi necessário recorrer a uma biblioteca de *JavaScript* para que, ao exportar os dados apresentados no *Excel*, estes fossem coerentes com os apresentados nas tabelas.

O último desafio, enfrentado nesta fase de implementação, foi um desafio de *performance*. Uma vez que o espaço da *cache* do *Mondrian* é limitado e as *queries* demoraram algum tempo a executar, decidimos usar apenas uma *query* para alimentar ambos os gráficos presentes no *dashboard*. Isto foi possível, porque, no gráfico à direita, apenas é mostrado um subconjunto dos dados apresentados que se encontram no gráfico da esquerda. Para isso, colocámos a mesma *query,* como fonte de dados de ambos os gráficos, e, no gráfico da direita, filtrámos os resultados, com base no último ano selecionado nos parâmetros laterais.

5.5. Integração dos vários módulos

Um dos requisitos do projeto é a integração dos *dashboards* que, posteriormente, vão ser integrados nos vários módulos que integram este estágio e que foram elaborados pelos restantes elementos da equipa. Uma vez que, à data da escrita do presente relatório, os *dashboards* do módulo aqui implementado ainda se encontram em fase de validação de resultados, não foram colocados os endereços dos *dashboards* implementados pelos colegas da equipa. No entanto, assim que a fase de validação de valores terminar, serão acrescentados ao menu de topo entradas com os endereços necessários para que seja possível aceder aos módulos dos colegas. Será ainda construída uma página inicial onde estarão imagens dos gráficos e endereços para cada um dos módulos implementados pelo grupo de trabalho.

Ao nível da *data mart*, optou-se por não fazer uma integração do modelo de dados, devido à complexidade de manutenção dos processos de *ETL,* dado que cada módulo teria o seu processo de *ETL* para atualizar as tabelas comuns, podendo dar azo a erros e inconsistências, durante a atualização dos dados.

Capítulo 6
Validação de Resultados

Neste capítulo, é apresentada a metodologia de validação da correção de resultados apresentados nos *dashboards* e, consequentemente, dos processos de *ETL* e das funcionalidades aí implementadas.

Validação de processos *ETL*

A primeira verificação, feita nos processos de *ETL* apresentados, foi a verificação de que todos os registos necessários estavam a ser transferidos do Nónio para a área temporária. De forma a garantir que as bases de todo o trabalho que iríamos desenvolver a seguir se encontravam sólidas, verificou-se que a soma dos valores em bruto e o número de registos na área temporária era concordante com os existentes no Nónio.

De seguida, verificámos se os pontos de controlo do processo de *ETL* estavam a ser realizados, através da definição de vários intervalos. Depois da extração de cada um deles, averiguámos, mais uma vez, se o número de registos e a soma dos valores extraídos eram concordantes com o mesmo intervalo no Nónio, para, eventualmente, não perder dados, quer devido à ocorrência de um erro na execução do processo *ETL*, quer no caso em que os dados seriam extraídos por intervalos. Estes dois primeiros passos foram realizados várias vezes até ambas as condições serem satisfeitas.

De forma a mitigar erros nos cálculos dos indicadores, decidimos adicionar às tabelas de facto do modelo de dados as chaves primárias das referências e das parcelas de compensação, como sendo dimensões degenerativas, ou seja, chaves que não têm relação com qualquer dimensão. Deste modo, nos processos de *ETL*, responsáveis por atualizar as tabelas de facto, conseguimos garantir que uma referência ou parcela de compensação aparecesse apenas uma vez para cada combinação das dimensões. Após termos executado o mesmo processo várias vezes seguidas, com o mesmo conjunto de dados, na tabela de factos, não apareceram quaisquer duplicados.

A verificação da correção dos cálculos efetuados nos processos de *ETL* será feita indiretamente, através da validação dos valores apresentados nos *dashboards* até que se encontrem dentro de um intervalo de valores aceitável.

Validação dos cubos

Para verificar se os cubos especificados se encontravam corretos, após o *deploy* para o *Mondrian*, no servidor *OLAP* e com a ajuda da ferramenta *Saiku*, que permite listar as dimensões e os factos presentes em cada cubo, foi feita uma inspeção visual das dimensões e dos factos especificados no modelo de dados e das dimensões e dos factos apresentados no cubo. Esta ferramenta permitiu, ainda, executar *queries MDX* nos cubos, o que possibilitou verificar se as operações de agregação estavam bem especificadas, com base no resultado obtido da *query MDX* e da correspondente *query* em *SQL*.

Validação dos valores e dos *dashboards*

Para todos os *dashboards* implementados, foi feita uma validação dos requisitos funcionais, numa primeira fase, e, numa fase posterior, foi pedida aos elementos do grupo de trabalho a sua colaboração na validação dos requisitos gerais do projeto. Assim, efetuou-se uma validação para verificar se todos os indicadores se encontravam implementados e, em cada *dashboard*, se todos os requisitos funcionais gerais estavam implementados, de acordo com o

especificado. Após esta primeira iteração, solicitámos aos colegas do grupo que verificassem se os requisitos funcionais gerais se encontravam todos especificados e se o seu comportamento se encontrava como tinha sido definido na análise de requisitos. Foi obtido um *feedback* de que havia algumas imprecisões que, neste momento, já se encontram corrigidas. Os requisitos não funcionais foram sendo validados, ao longo de todo o desenvolvimento do projeto, na medida em que todos os processos de *ETL* foram executados em sistemas *unix* e, para visualização dos *dashboards,* foram testados os *browsers* mais usados: *Internet Explorer*, *Firefox* e *Chrome*.

Após esta fase de validação, foram contactados os funcionários, Nuno Patão do Serviço de Gestão Financeira e Lucinda Abrantes da Unidade de Tesouraria, incumbidos de realizar a validação dos valores apresentados e das funcionalidades dos *dashboards*. Para esta fase, além de termos enviado o documento de requisitos, produzimos, ainda, um outro documento com uma demostração, exemplificando como navegar nos *dashboards*, e com valores obtidos para alguns indicadores. Neste momento, encontra-se já uma reunião agendada para a obtenção do *feedback,* relativamente às dificuldades de usabilidade e à correção dos valores apresentados. O documento produzido especificamente para esta fase encontra-se no anexo 1.

Capítulo 7
Metodologia e Planeamento

Neste capítulo, é apresentada a metodologia de desenvolvimento utilizada para o desenvolvimento deste projeto e as atividades levadas a cabo, ao longo de todo o ano.

7.1. Metodologia de Desenvolvimento

Apresenta-se, na figura 23, um diagrama da metodologia de desenvolvimento adotada, que corresponde àquela que foi proposta por Ralph Kimball, no livro *The Data Warehouse Lifecycle Toolkit*[15]. Adotou-se esta metodologia, porque o seu autor é uma referência na área de inteligência no negócio e porque ele próprio a usou, com sucesso, em projetos onde participou.

Figura 23 – Metodologia de desenvolvimento adotada

A primeira fase trabalhada, durante o primeiro semestre, foi a da definição de requisitos, que corresponde aos artefactos apresentados no capítulo número três, onde é feita a identificação dos requisitos do projeto e se destaca o levantamento de indicadores referentes a propinas e emolumentos e o levantamento dos requisitos funcionais e não funcionais.

Foram ainda trabalhadas as fases do modelo dimensional, físico e de desenho da área temporária, tendo sido apresentados os artefactos correspondentes a estas fases no capítulo número quatro, onde foi exposta a arquitetura do sistema. As fases de desenho da arquitetura e especificação da análise *OLAP* foram também trabalhadas, destacando-se para esta última os protótipos desenvolvidos e validados pelos clientes. Por fim, foi, ainda, feito um levantamento das tecnologias a usar no desenvolvimento do projeto, que corresponde à fase de seleção de produtos.

Durante o segundo semestre, iniciou-se a fase da instalação e configuração do *software* selecionado, trabalhou-se a implementação do modelo físico e da área temporária, desenvolveu-se a aplicação analítica e foi feito o *deployment*. Todas estas fases foram detalhadas no capítulo número cinco. A fase de validação de resultados foi explicada no capítulo número seis. A fase de manutenção e crescimento será assegurada pelo *GSSIC*.

7.2. Metodologia de Trabalho

Para a realização das tarefas, foram efetuadas reuniões semanais onde se encontravam presentes o orientador de estágio, os colegas responsáveis pelo desenvolvimento dos restantes módulos e o Engenheiro Pedro Pinto, na qualidade de orientador do módulo do sucesso escolar. Nestas reuniões, para além de terem sido atribuídas tarefas e de se exporem os problemas sentidos na realização do projeto, houve sempre, da parte do orientador, disponibilidade para nos ajudar a resolver as dificuldades apresentadas, tendo acompanhado sistematicamente o desenvolvimento do projeto e dando sugestões. Um facto também importante e enriquecedor foi a partilha de opiniões com os colegas de trabalho, na tentativa de solucionar as dificuldades sentidas. O acompanhamento das tarefas pôde ainda ser efetuado, através do *redmine*, uma plataforma *web* que visa gerir projetos. Para que a partilha de informação fosse mais eficaz e produtiva, usou-se a *Dropbox,* como repositório de informação, e, como forma de contacto, o *Skype*, o correio eletrónico e o *Facebook*.

Sempre que necessário, foram realizadas reuniões com os Engenheiros Pedro Pinto e João Amado para a especificação das vistas com os dados necessários para o desenvolvimento deste módulo. Dado que o local de trabalho corresponde ao mesmo espaço onde se encontra a equipa de desenvolvimento do Nónio, existiram ainda contactos informais, para explicações dos dados presentes no sistema e da lógica do negócio que está representada no modelo de dados do Nónio.

7.3. Planeamento

Antes de descrever as atividades desenvolvidas, no decorrer deste primeiro semestre, e de apresentar o respetivo diagrama de *Gantt*, salientamos o facto de o campo de acção deste estágio ter sido alterado, no final do mês de outubro, o que obrigou a que tivessem de se refazer algumas tarefas já realizadas ou iniciadas até ao final desse mês. Assim, na figura 24, a parte correspondente ao módulo económico-financeiro pretende mostrar as tarefas desenvolvidas inicialmente. O estado da arte, que tinha sido iniciado neste módulo, pôde ser reutilizado. No anexo 2, é apresentado todo o trabalho desenvolvido até ao final de outubro.

Dado que houve a mudança de âmbito, quase todas as tarefas, que estavam previstas para o primeiro semestre, tiveram de ser realizadas entre novembro e janeiro. Esta alteração originou atrasos no desenvolvimento do estágio, especialmente durante a fase de documentação de requisitos e prototipagem, devido ao facto de ser necessário integrar este módulo nos restantes módulos, da responsabilidade dos elementos do grupo de trabalho.

Para minorar as dificuldades referidas, revestiram-se de grande importância as reuniões levadas a cabo que ajudaram a redefinir as tarefas a desenvolver

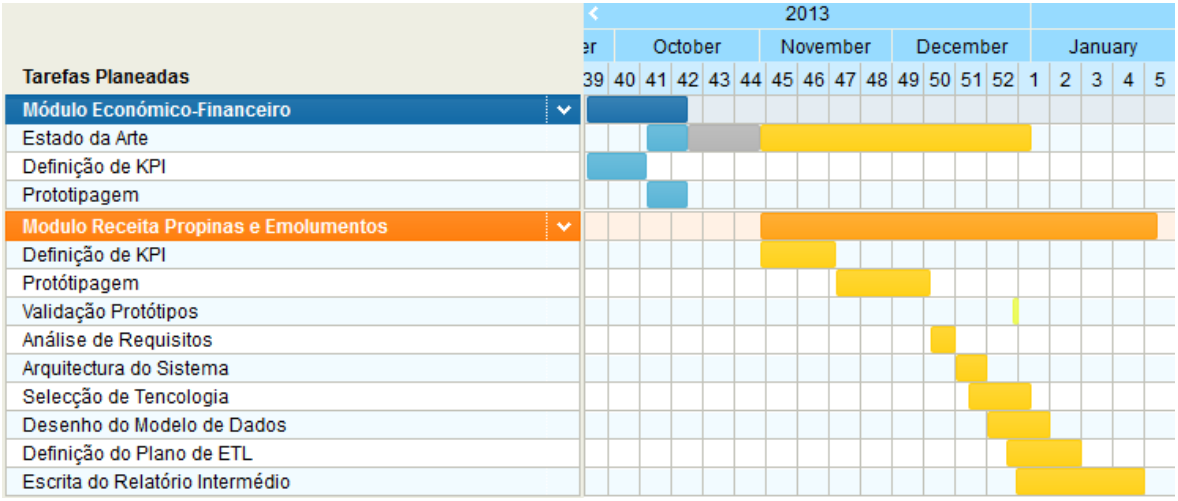

Figura 24 – Diagrama de *Gantt* para o 1.º semestre.

Na figura 25, apresenta-se agora o planeamento para o segundo semestre, sob a forma de diagrama de *Gantt*.

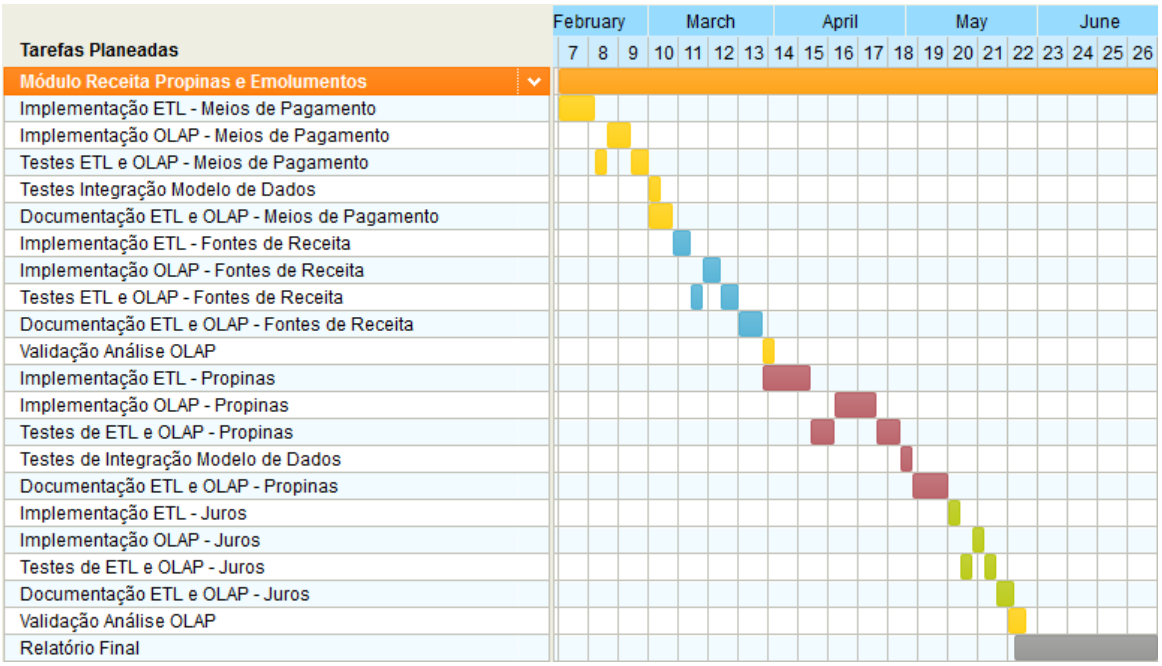

Figura 25 – Diagrama de *Gantt* com as tarefas planeadas para o 2.º semestre.

Na figura 26, é apresentado o diagrama de *Gantt,* de acordo com as tarefas desempenhadas e o tempo despendido em cada uma delas.

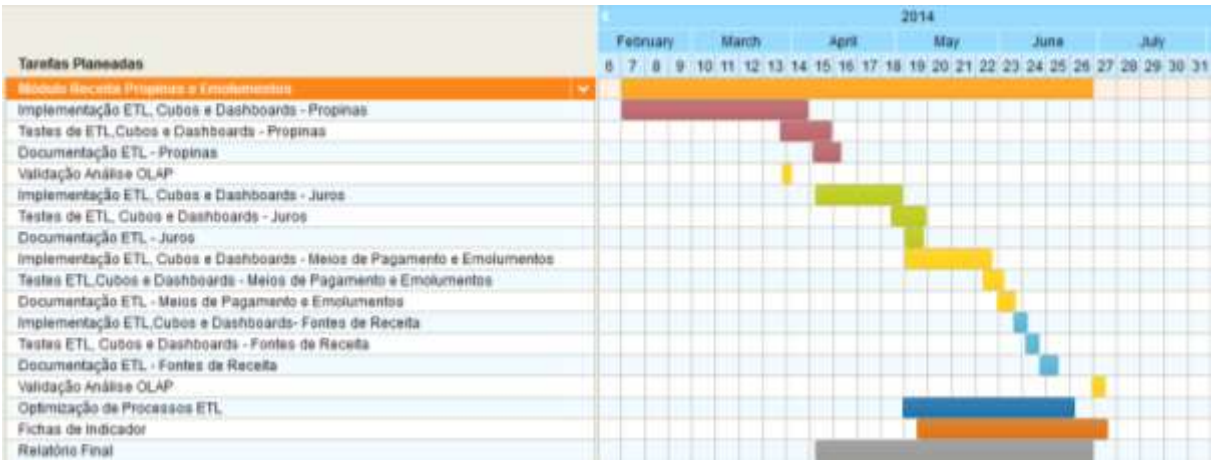

Figura 26 - Diagrama de *Gantt* das tarefas realizadas no 2.º semestre.

Em relação ao plano inicialmente definido, houve algumas alterações significativas. A primeira alteração foi a adição de novos indicadores, relativos à receita de propinas. Este acrescento fez com que este módulo tivesse uma prioridade maior e, consequentemente, tivesse de ser implementado antes de qualquer outro. Por se terem adicionado mais indicadores, a implementação do módulo foi alargada, em relação ao inicialmente estabelecido. Durante o desenvolvimento do módulo, houve alguns atrasos relativamente ao planeamento inicial, devido aos *bugs* que iam aparecendo durante o desenvolvimento dos *dashboards*. Não havendo muita documentação sobre o *plugin* usado, para solucionar os *bugs*, recorreu-se à escassa documentação existente, ao fórum do *plugin* e aos restantes colegas constituintes do grupo de trabalho. A implementação do módulo de propinas sofreu sucessivos atrasos, em relação ao plano inicial, o último dos quais está relacionado com a problemática da definição dos cubos virtuais. A solução encontrada para resolver o problema está descrita no capítulo número cinco.

Uma vez implementado um conjunto de processos de *ETL*, cubos e *dashboards*, o desenvolvimento dos restantes módulos ficou mais facilitado, dado que alguns dos desafios enfrentados, ao longo da fase de desenvolvimento, já tinham sido ultrapassados, durante o desenvolvimento do módulo de propinas.

Neste semestre, procedeu-se ainda ao desenvolvimento e ao preenchimento das fichas de indicador, que contêm, entre outras informações, a fórmula de cálculo dos indicadores, os níveis de granularidade, as fontes de dados, os filtros a serem aplicados e a periodicidade de actualização, permitindo ao GSIIC realizar a manutenção dos indicadores.

Capítulo 8
Conclusões e Trabalho Futuro

Neste capítulo, são apresentadas as conclusões, resultantes do trabalho realizado ao longo de toda a duração do projeto.

Em primeiro lugar, convém destacar que, no decorrer do primeiro semestre, foram cumpridas todas as tarefas planeadas. Foram definidos os indicadores a implementar, identificados os requisitos funcionais e não funcionais, com a ajuda de um protótipo rápido, e todos estes artefactos foram validados pelos utilizadores do sistema dos quais se destacam a equipa reitoral e o conselho de gestão. Foi definida a arquitetura do sistema, com base nos requisitos não funcionais, o modelo em estrela a implementar, bem como a área temporária necessária para apoiar o processo de *ETL*. Realizou-se, ainda, um estudo das tecnologias para cada uma das partes da arquitetura.

Durante o segundo semestre, mesmo com a alteração do planeamento e com a introdução de novos indicadores, as tarefas planeadas foram, igualmente, cumpridas. É de salientar que em alguns casos, foi demonstrado uma prova de conceito, como no caso da construção de agregados no modelo em estrela, não tendo havido tempo para os implementar todos até à data da escrita do presente relatório.

Como trabalho futuro, será essencial continuar o desenvolvimento de agregados, sempre que seja necessário, devido à natureza dinâmica da análise realizada pelos utilizadores do módulo desenvolvido. Esta tarefa deverá ser constante, durante todo o período de vida útil desta *data mart*. Devido aos atrasos mencionados no capítulo número sete, é ainda necessário realizar correções nos processos de *ETL* que deverão estar concluídas, nas próximas duas semanas.

Para concluir, podemos assegurar que este estágio constituiu um dos maiores desafios travados até ao momento. Podemos afirmar convictamente que atingimos o objetivo principal do estágio – melhorar os processos de consulta e análise dos indicadores da UC, relativamente à receita de propinas, de juros, de emolumentos e à utilização de meios de pagamento. A partir de agora, existe toda uma variedade de atributos que podem ser combinados para realizar a análise, o que se torna num factor diferenciador, comparativamente ao que existia até então. Ao nível das competências, o projeto contribuiu para elevar o nosso conhecimento técnico, no âmbito da engenharia de *software* e da área de inteligência de negócio.

Referências

[1] http://www.oracle.com/us/solutions/business-analytics/overview/index.html

[2] http://www.inetsoft.com/products/StyleIntelligence/

[3] http://www.sisense.com/

[4] http://www.softexpert.com.br/business-intelligence.php

[5] http://www.microstrategy.com/platforms/analytics

[6] http://www.sap.com/pc/analytics/business-intelligence.html

[7] http://www.jaspersoft.com/

[8] http://www.uc.pt/planeamento/PEA_2011_2015_out2012.pdf

[9] http://www.uc.pt/dpgd/doc_gestao/Relatorio_Gestao_Contas_UC_2012.pdf

[10] KIMBALL, R. 2002. The Data Warehouse Toolkit. The Complete Guide to Dimensional Modeling.

[11] http://community.pentaho.com/projects/mondrian/

[12] http://www.postgresql.org/

[13] http://www.highcharts.com/

[14] Silberschatz, A. 2006. Database System Concepts

[15] KIMBALL, R: 1998. The Data Warehouse Lifecycle Toolkit, New York

Anexos

Anexo 1

A. DOC_INDICADORES
B. DOC_ANÁLISE_REQUISITOS
C. DOC_ESPECIFICAÇÃO_PROTÓTIPOS
D. DOC_ESPECIFICAÇÃO_DESIGN_10-01-2014
E. DOC_ESPECIFICAÇÃO_PROTÓTIPOS_ALL_22-12-2013
F. DOC_VALIDAÇÃO
G. DOC_PROCESSOS_ETL
H. DOC_ANÁLISE_TECNOLÓGIA
I. DOC_FICHAS_INDICADOR

Anexo 2

A. DOC_KPI
B. DOC_MOCKUPS