

Business Intelligence System

BSc (Ordinary) Information Systems / Information Technology

DT249 Stage 3

Final Year Project

System Requirements & Design Document

Andrei Gurguta

**December 2016**

Table of Contents

[1.Introduction 3](#_Toc469266869)

[2.The Business Need 3](#_Toc469266870)

[2.1 Business Context 3](#_Toc469266871)

[2.2 The Problem 4](#_Toc469266872)

[2.3 The Solution 6](#_Toc469266873)

[3.Systems Requirements and Design 7](#_Toc469266874)

[3.1 System Viability 7](#_Toc469266875)

[3.2 User Stories 9](#_Toc469266876)

[3.4 Architecture 10](#_Toc469266877)

[3.5 UML Use Case diagrams 11](#_Toc469266878)

[3.4 Class and Sequence Diagrams 14](#_Toc469266879)

[3.5 Entity Relationship Diagram 15](#_Toc469266880)

[4.Technology Selection 16](#_Toc469266881)

[4.1 Options 16](#_Toc469266882)

[4.2 Conclusion 18](#_Toc469266883)

[5. BI Systems market research 18](#_Toc469266884)

[5.LO-Fi Prototype 20](#_Toc469266885)

[6.Project State, future work and risks 23](#_Toc469266886)

## 1.Introduction

This document is meant to describe the business utility and system requirements for the Business Intelligence System (from now on BI System) and to compare technologies/implementation options. Furthermore, it will describe the chosen system architecture as well as the design of its components.

As a high-level description, the BI System in scope of this document is a client-server type of software program that aims at resolving data integration problems from an analytical point of view, for small to medium size business organizations. The system will provide a pragmatic method of integrating the RDBMs of other business systems within an organization (e.g. CRMs, HRMs, Financial System etc.) into a single unified abstraction. In a more detailed level, this is a Federated and Virtualized business intelligence system.

To best reach the purpose of this document, the introduction regarding high level system requirements and business contexts that justify the utility of a BI System are verbalized from the perspective of a company named Clarity Ltd. This is an Irish SME which at its basis specializes in Assets and Office Management and is a small to midsize company with less than 100 full time employees. The names and other details presented throughout the document regarding business activities of Clarity or actual actors have been changed/anonymized in order to protect the identity of the real-world company on which the description is based. Nevertheless, the facts and business realities on which business and system requirements are based on, rely on a real-world entity encountered throughout the professional experience of the author.

## 2.The Business Need

### 2.1 Business Context

Clarity Ltd started as a small company approximatively 15 years ago by Carl, which took on a setup and maintenance contract of a major offices building in Dublin. Initially their purpose was to fit the building with electrical installations and equipment, ranging from air conditioned machines, water dispenser, and even electric bulbs. Also, they insured first hand[[1]](#footnote-1) maintenance and service for any of the devices they installed.

The company quickly registered success and in just a couple of years they evolved into offering even more products to be installed and even a wider range of services. Their crafty technicians combined with a stellar customer care resulted in more contracts with multiple offices buildings.

While initially they installed and worked only with the customer’s machinery, they went on with buying their own electrical equipment to resell, thus developing a stock of products. Even more, their range of services grew to offering more specialized maintenance and repairs to complex machinery. This meant they had to hire or subcontract more technicians. With time, Clarity organically grew to 40 permanent staff in different business areas like finance, payroll, admin, technicians, sales folks, customer service and so on. Also, the company maintains a network of 50 skilled contractor technicians to be dispatched on site.

In parallel this meant they had implement different IT systems to manage all of their business operations, personnel and financial matters. Initially they started with a very simple MS Access application to track their work orders, sales figures and business partners’ details, but with time it was obvious they had to extend this to more professional software. So as the business was growing they ended up holding a consistent number of business systems. Throughout the years they’ve implemented a WFM[[2]](#footnote-2) tool to manage on site work, a CRM[[3]](#footnote-3), an Accounting/Sales/Payroll system, a HR management tool, a stock management software and even a website, each of them holding its own database. Naturally, as the systems grew in number and complexity, Clarity had established a small IT department to manage these systems.

Linette is the head of this IT department which is responsible for the good functioning of 6 IT systems. Besides operational management of the software systems, Linette’s team is a gateway to the data repositories that each system holds in the databases. The accounts and payroll folks are her main internal customer to which she sends monthly data extraction files to be processed and compiled accordingly into financial statements and reports. Furthermore, management people like Carl frequently ask for cost/income data to further compile their own business performance analysis. So Linette and her handful of people in the department play a crucial role in the data management side too.

Coming now to our time era, Carl, which now is Clarity’s Chief Executive, decided it’s time to leverage the company’s notoriety and look into other business opportunities. So they proceeded with getting a bank loan to acquire two new companies and extend business verticals. The first is HomeWorks ltd which is a small company offering home repair services to house hold consumers. Its business is somewhat similar with Clarity’s in the sense that they both offer technician’s services mainly through professional contractors and have a dispatcher service to assign individual pieces of work. The second company to acquire is Cleaner Offices, a company which offers professional cleaning services to offices all over Ireland. With this second acquisition, Clarity aimed at having an almost complete list of services to offer to customers and also to extend their business geographically.

This acquisition created a new challenging situation for Clarity, beside the excitement of almost doubling the head count: two other sets of business systems were added to the existing collection that Linette and her team manage.

### 2.2 The Problem

This expansion made life more complicated for quite a range of people in the head office of Clarity from a data management perspective. Linette and the team had struggled to merge a few of the software tools and looked at even acquiring a smaller ERP for a single system approach. But after thoughtful analysis it was proven that the costs of acquisition and further customization of an ERP at this stage would be too much giving the variety of current systems and business perspectives. Linette and her team did a great job of ensuring good master data management and governance[[4]](#footnote-4) between systems but providing sensible unified data on a regular basis to the business consumers became something of an ordeal, as it will be further explained.

From the business side, Carl is more concerned with the profitability of the company given the loan. He is more demanding on profitability info, and is asking for more frequent reports on different business states. Steve, the financial controller of the company feels this pressure and is pushing himself the limits of the accounts department to provide the monthly financial statements at a faster pace. But this is proving to be more and more challenging. For example, before the expansion he could close the monthly accounts in a month end + 3 days but this period grew up to +10 days. Reconciling accounts payable, receivable, invoices, contractor billing and payments is becoming more complicated with the numerous sources of data.

Steve’s revenue recognition analysts now have to accurately compile dozens of large spreadsheets coming from Linette’s department. Bob, the senior financial analysts/accountant used to be the monthly recipient of about 10 data files that IT sends or that he himself downloads from the internal systems. With the increase in the number of system the number of data files he receives exploded to 35. These files contain sales, HR and billing data necessary to compute reports and financial analyses re the individual performance and bonuses of sales consultants. To make things even more complicated, due the nature of Clarity’s operations, Bob’s team is in charge of making sure that whatever is being payed to the contractors is billed to the customers. Any delay’s in accurate billing can have an impact on the company’s figures.

Besides Steve and Bob in the financial side of the business, Margie which is in charge of managing Clarity’s stock of electrical machinery, is dependent of some extensive reports from the Stock Management system but also on some data extractions and reports from the company’s web site to cross-match available offers from suppliers with current stock. Clarity now holds 3 websites thus tripling the effort of combining data for Linette’s team.

I have put some background on the different departmental data needs to better explain the data “paralysis” that Clarity is experiencing. All departments are in demand of more integrated reports and complex data extracts, and they are all pushing back to IT. Before the expansion, Steve, Bob and Margie were perfectly capable of creating their own reports either straight form the system’s interface of by asking Linette’s team for ad-hoc queries from the system’s databases. They would mainly process raw data files from IT thanks to their excellent Excel skills and up to a point managed data integration by themselves.

Now the situation is different and IT are dealing most of the time with more reporting requests rather than ensuring systems availability and development. Many of the team’s resources spend most of time developing complex queries against a set of 10 system databases and compiling these into unified files. Business questions are becoming more and more diverse so just saving the SQL queries and re-running them isn’t an option any more.

Linette sees that the situation has dramatically changed in Clarity and acknowledges that business users have a hard time in getting a unified view of the data held in the different systems. She realizes that the business would greatly benefit if there was a single logical place for data and reports to be consumed from. Even for the Excel savvy people in finance it would be a great help if they could download unified datasets from a repository based on systems integrated data directly in their spreadsheets. This could really boost productivity for both sides, relieving some pressure off the IT’s shoulders! Linette envisages the business case for a Business Intelligence System so she addresses Carl and the management team.

### 2.3 The Solution

Linette presents her view to Carl and the rest of the management team and they all agree that the expansion of the company effectively transformed Clarity into a data driven company. Valuable information is scattered throughout the independent system on which Clarity operates. Something has to be done to get a wider integrated view of data from all systems in a timely fashion and everybody understands that delivering combined data sets or developing ad-hoc reports is consuming too much time for IT.

Carl also loves Linette’s idea of having a single logical place, maybe a website, where he could get reports on demand and not wait for them to be sent by email. Steve and the rest of finance also agree that having the possibility of downloading more integrated data directly into their spreadsheets would be of great help. Maybe thus they could go back to closing accounts in a month +3 days fashion again.

Having the consensus of all company, Linette proceeds in analyzing the best technical solution for the problem. She initially looks at maybe building a data warehouse. But soon realizes that this would be too costly for a company of Clarity’s size, given the additional licensing for a database software, ETL tools, storage costs and even specialized IT headcount to develop and maintain yet another system in the company. She also takes a look at some Cloud solutions but again feels that it would be too expensive and not everyone would be so comfortable with having the company’s data in a remote data center.

Throughout her research she sees different reporting tools and OLAP systems but neither of them makes sense for Clarity in Linette’s opinion, as they all involve basically developing and maintain a physical data storage with increasing IT’s headcount.

Then she encounters the concept of Federated and Virtualized data. This is somewhat of a new idea highly coupled with the concept of an Agile development environment where data views are incrementally build and ultimately the business users are empowered to consume data from multiple sources at their own pace.

The concept behind BI Systems that implement data Virtualization is quite simple from an IT professional’s perspective. Linette’s department would only have to setup a server with multiple connections to all available system’s databases then write and save SQL queries to be run on demand against these sources, thus producing a logical data view for each query within the BI System. Such views would be highly user friendly for business people as they would never have to write an SQL query.

For example, Linette could write and save a query to fetch all sales invoices for the current month from the AR system then save the resulting data view in the BI System as “Revenue”. Then she could develop and save a query with all sales employees and their bonus targets from the HR systems (plus details) then save the resulting data set as “Sales Targets”. Business users could consume these data views from the BI system directly by simply demanding the data.

Furthermore, and this is where this type of system is advantageous in Linette’s perspective, at the BI System level these data views can be joined and combined to produce other meaningful views. In the above example the 2 data sets could be combined and aggregated by the sales employee ID to get an accurate perspective on employee achievements for the current month. So under the hood, the BI System would initially run the 2 base queries then joining them in the application layer to publish them as 1 integrated view of the 2 database system.

Linette quickly sees great the potential of this approach. Building views in an incremental manner would greatly speed up development of reports and data extracts. Because the views can be constructed one on top of the other right in the BI System layer with no SQL coding skill could mean that the users them self can build them at some point. This would mean a more rapid validation of the results by the consumer himself so massively reducing the need to go back and forth to the IT department.

Then all the developed data sets can be consumed through a UI that the BI System offers. Business consumers could then use their browser to navigate and view the created data sets, off course having in place a security policy. Even more, such a BI System is very extensible by offering the possibility of consuming the federated data as a web service. This means that the finance people can hook up their spreadsheet applications directly to the BI System and use them further in local computations.

This all sounds great for Linette so she puts down on paper what would be the costs and man power to run such a system. Broadly, that would be the system software itself, the hardware could be a simple commodity server as there wouldn’t be the need for a separate database or large storage where to gather the data. Regarding the staff, she would need initially someone to gain knowledge on how to administer the system, setup the database connections and develop the initial SQL queries. The potential is that in a while, after business users get accommodated with the BI System, they can create new Joined Views from the initial Base Queries by themselves.

Linette is satisfied enough that this is a good solution to Clarity’s current data paralysis and prepares her presentation for Carl and the rest of the management. She will propose the implementation of a Federated virtualization BI System!

## 3.Systems Requirements and Design

One of the reasons for verbalizing the Business necessity of a federated and virtualized BI System through an epic from Clarity’s perspective was that of introducing a few of the business actors who will benefit from and use the system. In the following section, the system’s requirements are written as a solution that targets resolution of The Problem as from these actor’s point of view.

The requirements are grouped into 2 sections, first describing the functionality and characteristics that make such a system viable for Linette and in analogy for any SME that falls in the profile of Clarity. Second, the document will materialize more specific functional requirements in “user stories” from the perspective of each actor.

### 3.1 System Viability

As a system’s manager, Linette is most concerned with the cost and manpower necessary to run the system. From her point of view, the easier the BI system is to setup and to use by all actors is then the better. So, to better print out the system’s viability (in terms of having the option of choosing between multiple implementations) requirements are stacked in 2 groups, first Resources and skills then Usability.

|  |  |
| --- | --- |
| BI SYSTEM VIABILITY | |
| SKILLS and RESOURCES | The BI System needs to make use of available physical resources as much as possible. It needs to be deployed and run on a commodity server which is manageable with available in-house administration skills. All systems in Clarity run on a MS Windows OS server[[5]](#footnote-5), so ability to be deployed and run on such an OS is the first path to be look to. |
| The system must be able to handle it self the processing of data or even better delegate it to the source system database. It should not incur the purchasing of an additional license for a commercial RDBMS engine for processing horsepower. |
| The skills necessary to run, maintain and operate the system must be in reasonable and reachable with existing in house IT staff. As mentioned above, skills to administer should be in the reach MS Windows servers. To operate the system only general IT knowledge should be necessary. To develop BI data views, only RDBMS and SQL coding skill are required. |
| The system must provide a graphical user interface to operate with and for the resulting data views to be consumed, either a web or a desktop one. The system must not require command line input or configuration files for day to day administration or use. |
| USABILITY | The BI System must be capable to connect and run SQL queries against traditional database engines (MS SQL Server, ORACLE or IBM Db2). |
| The base data views can be developed through standard SQL, native to each RDBMS. The system must not require additional proprietary languages or XML format commands to be learned. |
| The system must provide the functionality of creating additional views based on base views. These additional views must be computed within the system itself and not require additional software to be installed. |
| This second layer of views can be created by people having only general IT and data skills, but with non SQL coding. The development must be as simple as possible, through a wizard like interface or a graphical UI. |
| The system must provide for future extensibility and integration with other software systems and tools. So it The system must offer a way for the data to be consumed by other tools, maybe a web API. |
| The BI system must provide division of users functions. Users that administer and create base views in SQL must be delimited in capabilities than users that develop the 2nd layer of views and divided again from users that only consume data. This will allow for auditing and traceability processes to be put in place. |
| The BI System must provide 2 separated GU interfaces to adhere to implement the division of user function. It must have a separate management console and a separate UI for the data to be consumed through. |
| The system must provide a concept of security permissions to be put in place so that access to different data views by different business users is granted by layers of security, based on user permissions to different data views, or business functions (roles) |

### 3.2 User Stories

This section identifies the different features that the BI System is required to have in order to meet with the business need described in sections 2.2 and 2.3. They have been written in a first-person style from the perspective of different actors in Clarity, to emphasize who would be the main beneficiaries of each feature. The actors are IT admin, IT user and business user. Also there is one special case of business super user:

IT:

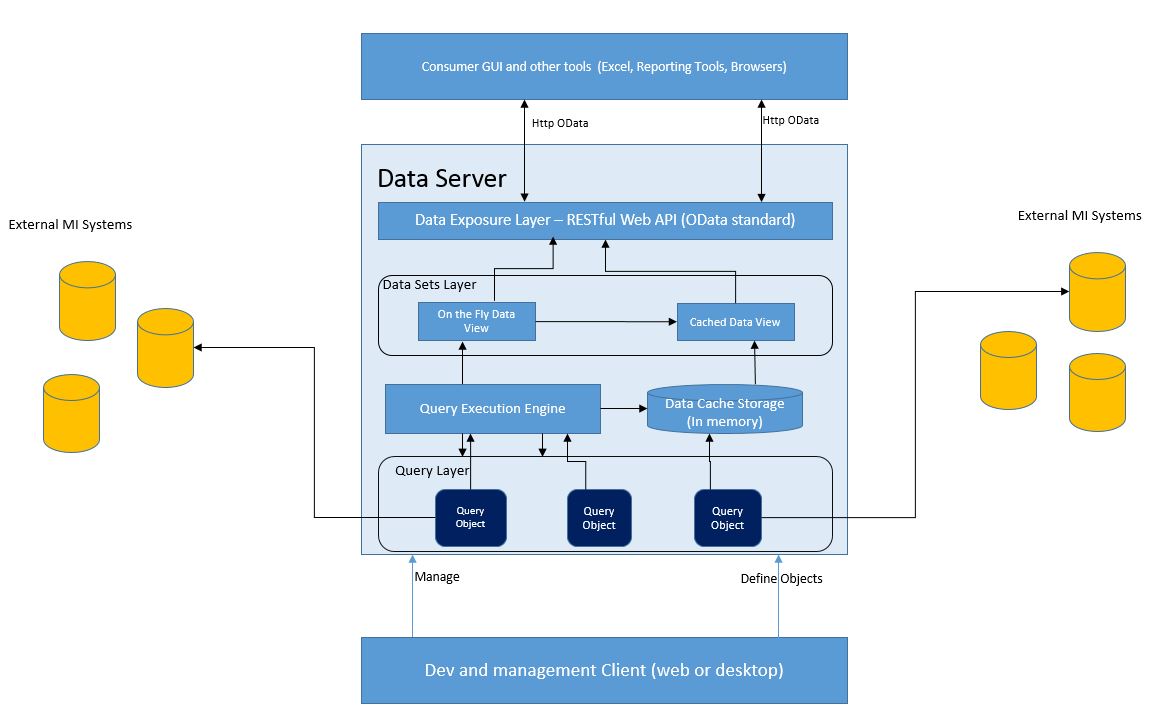
* As IT admin, I want to define users in the system and assign them to a user role/group
* AS IT admin I want to be able to grant different permissions to these roles/groups, in terms of what parts of the systems the users can access.
* As IT admin, I want to be able to create and store database connections as templates to be used in individual queries.
* As Business Manager I want to be able to control who gets access to different parts of the System
* As IT user, I want to define new data views (Joined Views) which will be based on at least 2 individual SQL queries (Base Views).
* AS IT user, I want to write/define SQL queries from within the system to implement these data views.
* AS IT user, I want to be able to test the SQL syntax of a query as I’m writing it in the BI System.
* As IT I want to develop queries that allow the use of parameters.

BUSINESS:

* As business user, I want to be able to open a data view and input parameters if necessary.
* As business super user, I want to create data views (Joined Views) on top of other views, by joining them on different columns.
* As business user, I want to be able view the result of a running data view through a UI
* As business user, I want to be able to download the content of the data view on my desktop
* As business user, I want to the downloaded data to be in a form of value delimited file
* As business user I want to be able to download the content of a data view directly in a spreadsheet application

Moving towards the implementation of the software system to meet the above business requirements, the following diagrams will describe, in a more technical manner, the main software components of the system and how users will interact with it. Each diagram is accompanied by a high level description.

### 3.4 Architecture

**General architecture diagram**

As the label suggests it, this diagram describes the general architecture and high-level components of the BI System. From the business requirements list it can be concluded that the implementation has to be a client-server architecture which will allow multiple users to access various features at the same time.

The diagram is dominated by the Data Server which is in fact the core engine of the BI system. In short, the main functions of the Data Server will be that of receiving user requests for data, resolving/dispatching underlying queries to be run at source DBs and also carrying out the binding of multiple query results into a single data view, if case. The Server itself will be logically composed of different interconnected components and subsystems, grouped in 3 layers.

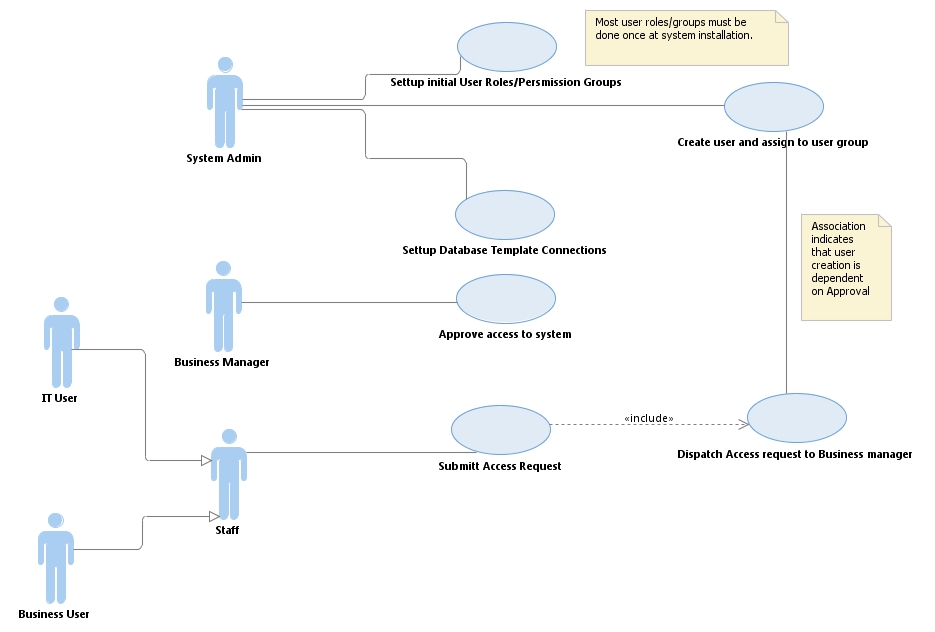
Starting from bottom-up in the diagram, Query Objects are the main parts of the Query Layer, having the role of storing the query text, parameters and other metadata for the actual SQL queries to be run against the operational data sources. Moving up to the Data Layer, the Data Views provide and abstraction of the Query Objects for the final consumer users. The link between the 2 layers is implemented by the Query Execution Engine which is responsible of resolving user requests at run time by actually running the SQL queries (Query Objects). The 3rd layer of the Data Server is represented by the Data Exposure Layer, which in short has the role of providing an interface (web api) for the data to be consumed with the imbedded GUI or by applications outside the Data Server.

A BI system wouldn’t suit its purpose of helping businesses leverage a wide range of disparate data sources if it wouldn’t offer a proper human oriented method for user interaction. This is the role of the 2 subsystems presented in the diagram, the *Dev and management Client* and the *Consumer GUI*.

The Dev and management Client, as the name simply states, is focused on the IT and power users by offering a graphical user interface for developing new views, SQL scripts and connections or to manage the day to day system usage. On the other side, the Consumer GUI is mainly directed to business folks, and it offers a way of displaying the consumable data integrated by the system. It will offer a catalogue-type perspective of all available data views for the user to choose from.

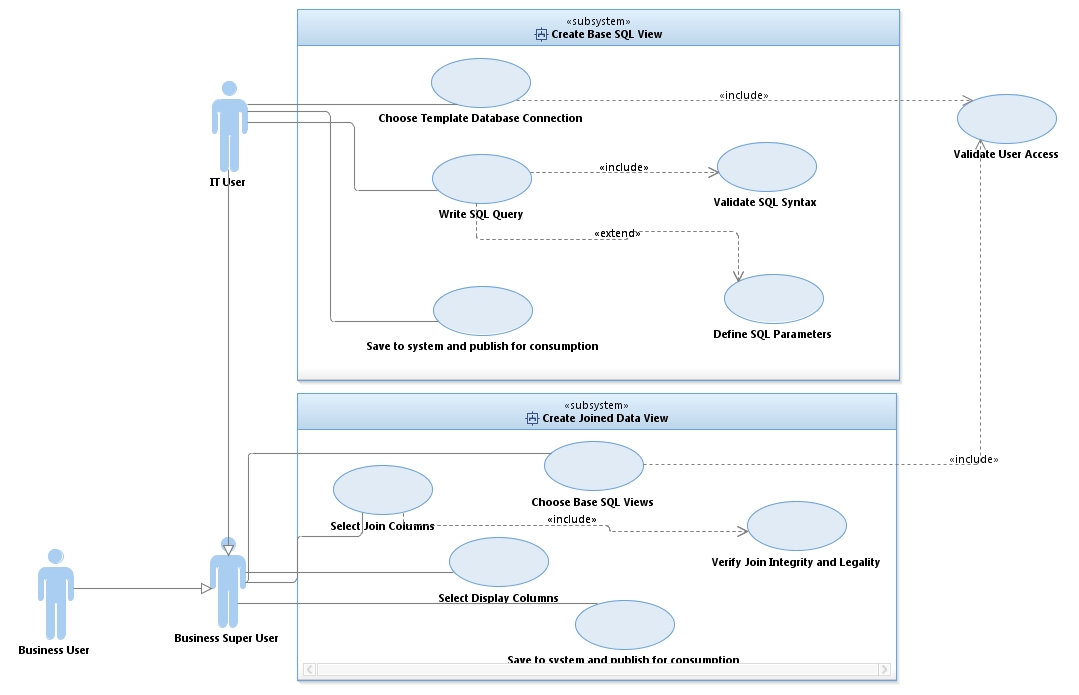
### 3.5 UML Use Case diagrams

The following 3 use case diagrams offer a logical perspective on some of the functionality and workflows that the BI system implements. The 3 use cases do not offer the full usability picture but only the main uses, that of System Setup and Permission management, Development of Data Views and finally the use case for consuming data.

1. **System setup and permissions**

This diagram mainly refers to security and administration workflows. Security, as mentioned in the Business requirements section, is a very important aspect for the business organization so well defined mechanisms must be implemented to make sure access to sensitive data is restricted and transparently controlled by business owners.

As per the diagram, the System Admin actor sets up Database Connections, creates Users and permission Roles/Groups. The Staff actor, which is a generalization of the IT or Business User actors, will submit access requests through the system, which in turn will have to be approved by the Business Manager actor. Based on this approval the System Admin will assign System Users to corresponding user roles depending on the access level granted.

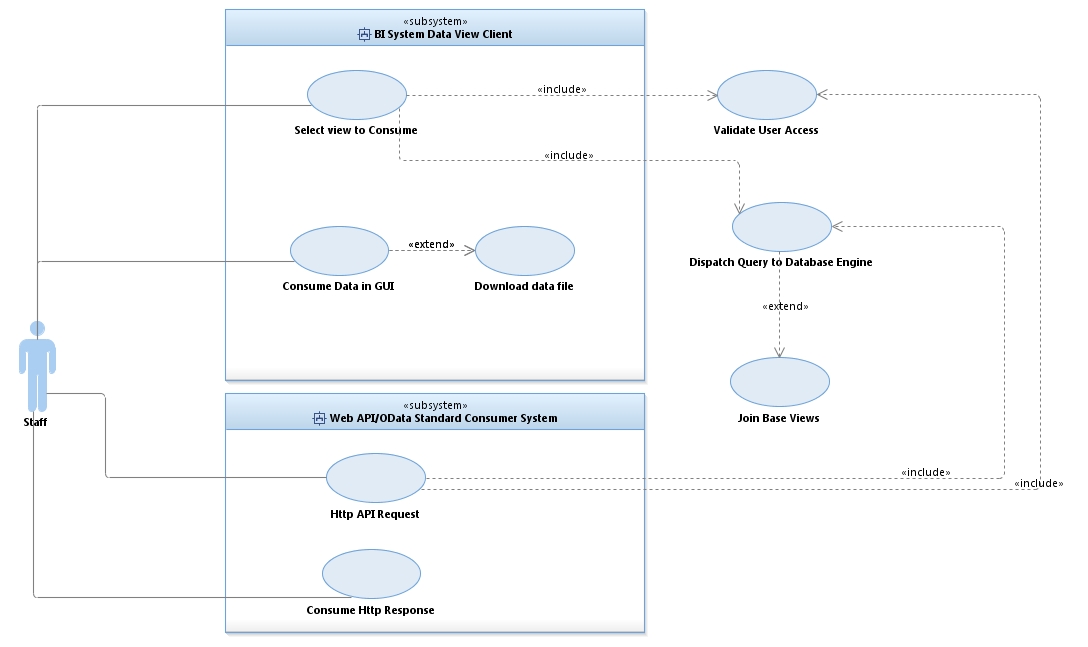
1. **Development of Data Views**

Use case B offers an overview of the workflow necessary to develop Data Views in the BI System. At the same time the diagram introduces the separation between the platforms necessary to instantiate the 2 different type of views: Base Views (SQL queries) and Joined Views.

In the first part, only the IT User is meant to interact and operate in the different scenarios for creating an SQL view. The logic behind this functional separation is due to the fact that SQL coding skill are necessary to develop the views, which will be mainly in the interest of this category of personnel.

Joined Views on the other hand, are meant to be created by Business Users too (via the Business Super User generalization) through a user friendly GUI (possibly Wizard alike) that the BI System has to offer according to the requirements.

The importance of the security mechanism the system has need to feature is again emphasized by the Validate User Access use case evoked in both subsystems.

1. **Consumption of Data Views**

The 3rd use case diagram is a representation of how the data can be published by the BI System. This can be accomplished either by a Graphical Client (Consumer GUI in the architecture diagram) or through the use of the Data Server’s WEB API by another tool (like Spreadsheet applications). The importance of security access validation is again specified.

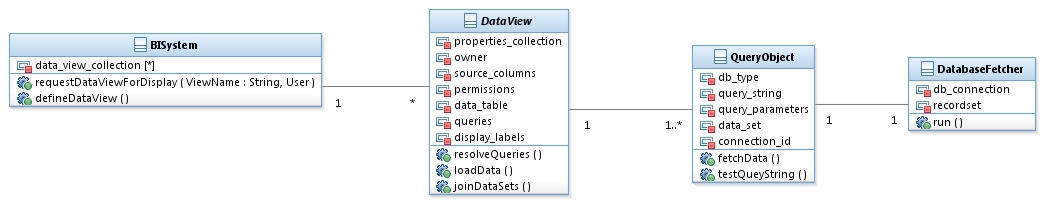
Also, another aspect which this diagram better illustrates is the workflow of selecting a view for display, which in turn dispatches the query to the operational database engine to be executed. This process will be described in more detail in next subsection.

### 

### 3.4 Class and Sequence Diagrams

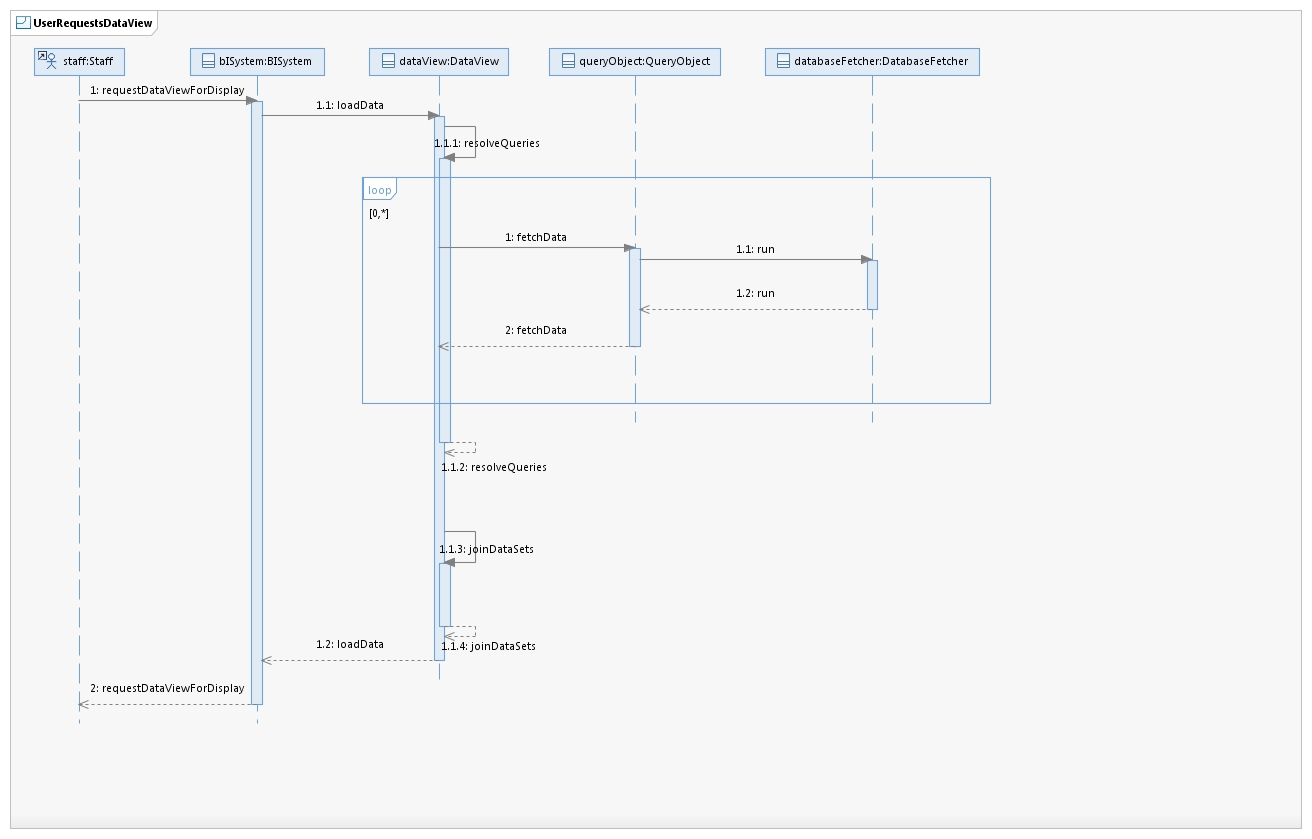
This subsection illustrates the core classes of the Data Server system and the relationships between them. Also, the sequence diagram shows the messaging model between the classes.

1. **Core Class diagram**



At the top of the Class diagram (or at the left better said), the BISystem is the logical container for DataView and acts as the facade entrance point for data requests. DataView is meant to be an abstract class for the 2 subtypes of views (Base SQL Views and Joined Views). QueyObjects are the physical implementation of SQL blocs that when being resolved by DatabaseFetcher objects will in turn return a data set to DataViews for further processing (if necessary).

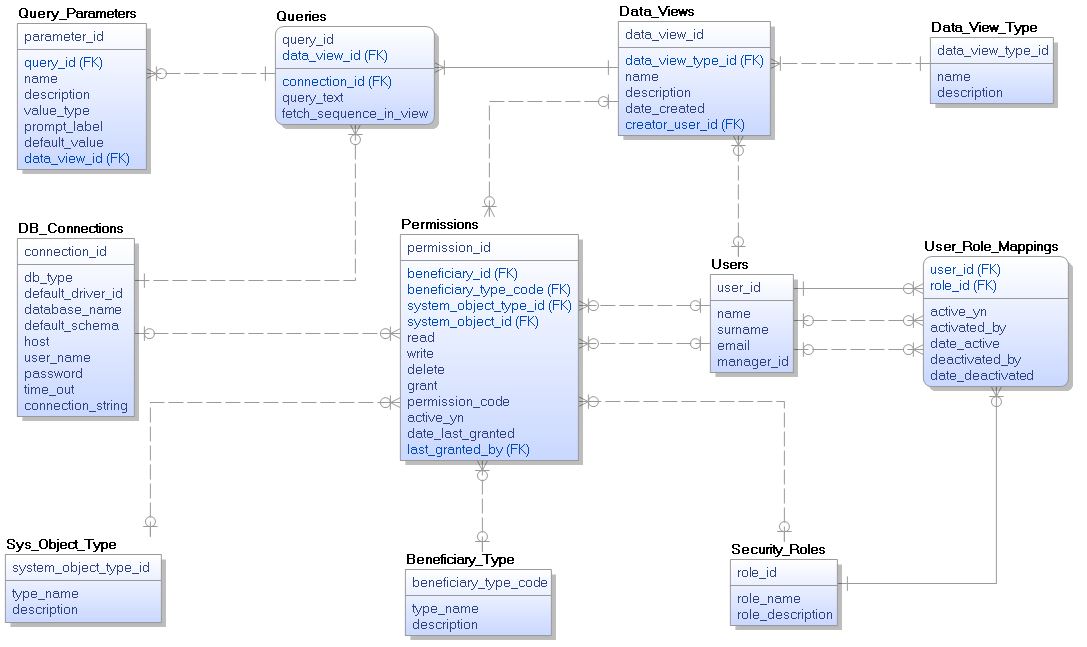
This logical separation between DataViews QueryObjests and DatabaseFetcher is meant to implement a loosely coupled system that allows further extension if necessary. All classes have a single purpose so DataViews don’t need to be aware of the actual physical implementation of the QueryObjects. On the same principle the QueryObjects don’t need to know how the data is physically retrieved from the data source, leaving the work of binding connection strings, parameters and DB drivers to the DatabaseFetcher.

1. **Messaging model**

The messaging model sequence represents one of the main transactional interactions between the classes presented above. Upon user request for data, the BISystem will ask for the corresponding DataView instances to initiate the data processing and preparation workflow. In this sense, the DataView object will message all of its referenced QueryObject instances to resolve themselves by dispatching their properties to a DatabaseFetcher.

DatabaseFetcher instances resolves the actual SQL execution and returns a recordset back the QueryObject with which it is binded. Having all QueryObject instance handling a recordset, the DataView prepares the data set and signals back to the BISystem instance (which be a singleton case) that data is ready to be consumed.

### 3.5 Entity Relationship Diagram



The software system built for the purpose of this project will be a virtualized and federated BI System, so data from the source databases won’t be stored physically on disk in a relational database. Still, being a software system it will need to store some of its persistent configuration/operational data structures. The best method in this case is a light weight RDBMS, such as SQLite or an express version of traditional commercial databases.

The ERD diagram above is the model of how different logical structures in the BI system will be persisted. Structures like Users, User Roles, Data Views, Queries or DB Connections plus other metadata information and relationships are modelled into normalized tables.

## 4.Technology Selection

This section is dedicated to the actual technology stack chosen for developing the BI System. If in the previous two chapters, the discussion was about the reason of why to implement such a system or system design and high-level architecture, in this next section the subject is the tooling necessary to implement the software. The following paragraphs will take into comparison different programming platforms by analyzing their pros/cons and in the end concluding on the best choice for this purpose.

### 4.1 Options

When researching over possibilities of implementing a client-server based software system, mainly 4 programming languages and platforms come in option: PHP, Python, Oracle Java and MS .NET.

1. **PHP** is a [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting) language designed primarily for [web development](https://en.wikipedia.org/wiki/Web_development) but also used as a [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language)[[6]](#footnote-6). It supports the whole set of object-oriented programming features and offers a rich library of extensions available as open source.

According to multiple opinions it is best used for developing web sites and there are many examples of successful web applications that use PHP including Facebook which apparently make use of this technology when speaking about front end development. Being a loosely-typed programming language one of its key advantages is that is quicker to learn and is considered to offer fast development times.

When speaking about developing a web services application or backend systems modules in PHP, many experienced developers would argue against this platform as first to choose from. Arguments in this sense vary from global configuration parameters that complicate deployment and portability, to scalability issues when talking about multithreaded execution.

1. **Python** is a widely used [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), [general-purpose](https://en.wikipedia.org/wiki/General-purpose_programming_language), [interpreted](https://en.wikipedia.org/wiki/Interpreter_(computing)), [dynamic programming language](https://en.wikipedia.org/wiki/Dynamic_programming_language). Its design philosophy emphasizes code [readability](https://en.wikipedia.org/wiki/Readability), and its syntax allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than possible in other languages[[7]](#footnote-7). One of its most important feature is syntax readability and the fact that Python programs are written in fewer lines of code, a fact known to result in fewer bugs. This makes Python also widely promoted as the best language to start learning programing. Furthermore, another key aspect of the platform is the extended python community which in turn results in a very comprehensive set of libraries. Python scores even higher rated reviews than PHP on speed of developing software, and is most often used in automating system tasks.

On the downside, the platform isn’t best regarded for its lack of true multiprocessor support and low-performance in multithreaded applications. Also, when analyzing the capabilities of Python in the area of database access layers and its ability to interact with powerful SQL engines, it is considered a primitive solution in comparison with established technologies as ODBC and JDBC.

1. **Java** is a general-purpose [computer programming language](https://en.wikipedia.org/wiki/Programming_language) that is [concurrent](https://en.wikipedia.org/wiki/Concurrent_computing), [class-based](https://en.wikipedia.org/wiki/Class-based_programming), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), and specifically designed to have as few implementation dependencies as possible[[8]](#footnote-8). It’s core feature is that of portability between OS platforms. This means that Java software can be written once on a given platform but it can run on all systems that support Java without the need recompilation.

The Java ecosystem has been present on the programming scene for more than 20 years, and when looking over the multitude of application domains still being developed in Java it is safe to conclude that it is continuously evolving. The platform benefits from a wide library that enables its successful implementation in client-server applications, enterprise, big data, mobile and many more. Technically speaking, one of Java’s strengths relies on stellar object-oriented features, that simplifies adding a great amount of abstraction to business logic in applications. Also, unlike Python, Java applications benefit from efficient garbage collection built in the platform.

These aspects combined with the high availability of skilled developers in this technology make Java one of the first languages of choice when embarking in a new application development project of more considerable sizes.

When discussing about downsides of developing in Java, there are also some aspects to be considered. First, because it runs on Virtual Machine, it runs somewhat slower compared to similar languages like C++. Also, Java is considered to add some development overhead, requiring more lines of code to develop small applications.

1. The final platform to be analyzed is Microsoft’s **.NET** and the **C#** programming language. The .NET Framework is a [software framework](https://en.wikipedia.org/wiki/Software_framework) developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) that runs primarily on [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows)[[9]](#footnote-9). It provides [language interoperability](https://en.wikipedia.org/wiki/Language_interoperability) meaning that each available language can use code written in other languages. Programs written in .NET execute in a [software](https://en.wikipedia.org/wiki/Software) environment known as the [Common Language Runtime](https://en.wikipedia.org/wiki/Common_Language_Runtime) (CLR) which can be best described as an [application virtual machine](https://en.wikipedia.org/wiki/Process_virtual_machine).

The .Net framework is known to offer top performance when developing systems that run on a Windows platform. It is seamless integrated with the OS and offers a rich library that greatly improves interoperability, deployment and reusability compared to most of the competition. Also, being a top grade enterprise platform, .NET provides great support for projects that involve data integration and intensive data operations. As opposed to PHP and Python, .NET offers a comprehensive solution for multithreaded programs. Furthermore, .NET code is compiled and is almost always going to run faster than the other two.

From a language perspective, C# has a C style syntax but offering a more mature set of constructs than Java. Examples such as formalized get-set methods, clean events management mechanism through delegates or even simplified multithreading control make C# a popular choice. One of the greatest assets of C# when developing data oriented programs is LINQ (Language Integrated Query) which is a declarative compact approach to modelling data queries within C# itself.

Coming to disadvantages of the .NET platform, the first thing to be kept in mind is that this is first targeted to MS Windows operating systems. There are some efforts of making the .NET more platform independent through projects like Mono or Xamarin but according to most benchmarking sites the run time performance is lower than on Windows systems. Furthermore, C# appears to not have the same benefit in libraries and available documentation compared to the Java platform.

### 4.2 Conclusion

After analyzing the 4 options, the main candidates to be chosen to develop the BI System come down to Java and .NET C#. Compared with PHP and Python which mainly target web sites and system automation development, Java and C# offer better features and support for developing client-server, multithreaded applications. Furthermore, they offer high quality documentation from Oracle and Microsoft plus an extensive set of libraries that best fit the purpose of developing a BI System.

When analyzing from the perspective of the business requirements, C# is the most reasonable choice, having a more promising support for backend data oriented programs. Built-in language features like LINQ would offer an easier way to code data operations modules. Furthermore, as per the System Viability description, .NET would be the best choice having the potential of becoming better integrated in the existing ecosystem of Windows based servers.

## 5. BI Systems market research

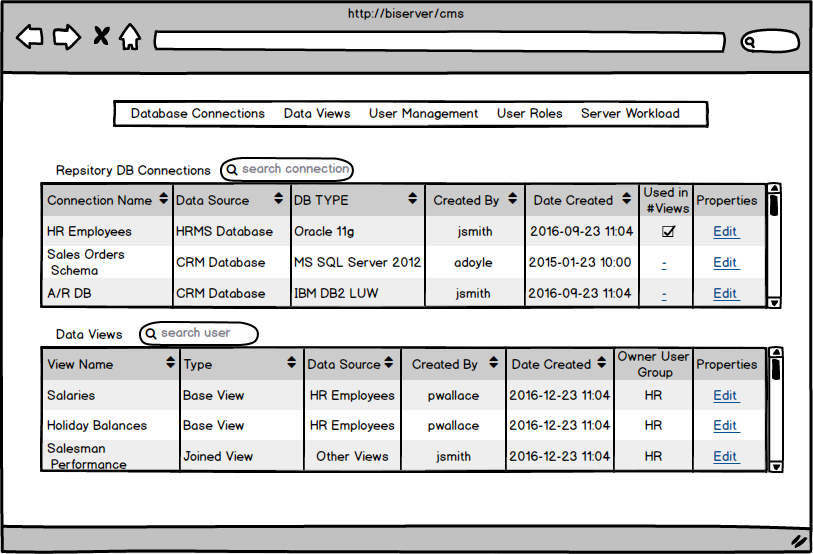
This chapter looks after making a study of existing BI Federated and Virtualized software products available on the commercial market globally. This is to prove that the proposed software architecture for the BI System subject to the degree project is still up to date in terms of how modern systems are delivering analytic capabilities to companies of small to medium sizes.

1. One of the most popular BI virtualization systems in North-America is CISCO Data Virtualization (from now on CISCO DV) formerly named CISCO Composite. This is a complete BI solution which offers out of the box data integration and self-service analytics through virtualization, as per the vendor’s description. The term used in the industry for such BI solutions is Agile BI, in respect to the promise that implementing one of these virtualization platforms is meant to deliver a faster ROI than physical storage approaches. At the core of CISCO DV stands the **Data Services** server, which is abstracted to consumers as the “Logical Data Warehouse” (LDW). Specialized users construct Business Views within the LDW which are meant to federate disparate databases from a business organization. Business users consume Business Views directly through the available GUI Clients or can hookup any 3rd party software to consume data through the available WEB API.
2. Another popular product, especially in mainland Europe, is SAP’s Business Objects BI Platform, the EDGE edition for SMEs (BOBI Edge). BOBI Edge is described by the vendor as a powerful BI choice for midsize companies who want to quickly leverage business data in order to gain a competitive advantage. The core backend data system is based on the same principle of federating and virtualizing data sources for easier consumption throughout the business organization. The BOBI server abstracts different areas of a business’s data into logical groups of views, named Universes. Compared to CISCO DV, this product has a more complex offering of components, including a comprehensive set of data consumer clients to create reports, analysis and data visualizations.
3. Moving on to the local software market, the Irish company Nathean Technologies offers a product named Logix Professional, which is a federation and virtualization BI system built on the same Agile BI principles as CISCO DV. Logix avails the Linx Server, which again offers an abstraction view of a company’s business systems’ RDBMs. Business users can consume data from this server through a built in GUI Client or can connect to it through a web API.

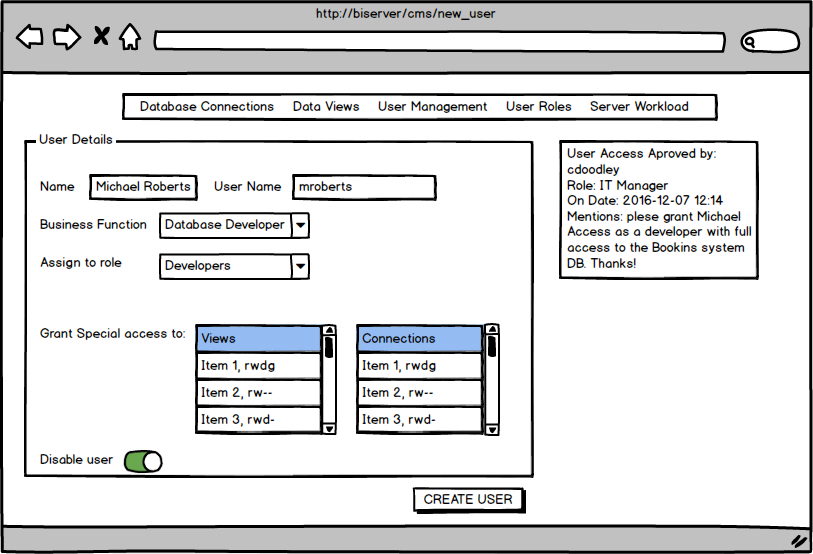
In conclusion, it is safe to assert that the proposed approach and architecture for the BI System is still up to date. Offering business intelligence to SMEs through data virtualization systems is a widely used technological solution. As with the commercial options described, the software system subject of this degree project, replaces the physical replication and remodeling of data existing in different RDBMs. Specialized IT users will create a set of Base data views, which can be extended by other power business users with Joined data views. Furthermore, data can be consumed by users using a built in GUI Client or with hookup a 3rd party software program capable of reading standardized WEB APIs.

## 5.LO-Fi Prototype

Moving a step closer to the incarnation of the project, this chapter aims to provide a zoomed-out idea of the usability of the BI System by listing a series of graphical user interface mockups. These mockups constitute the main workflow types that the system will make available to its users.

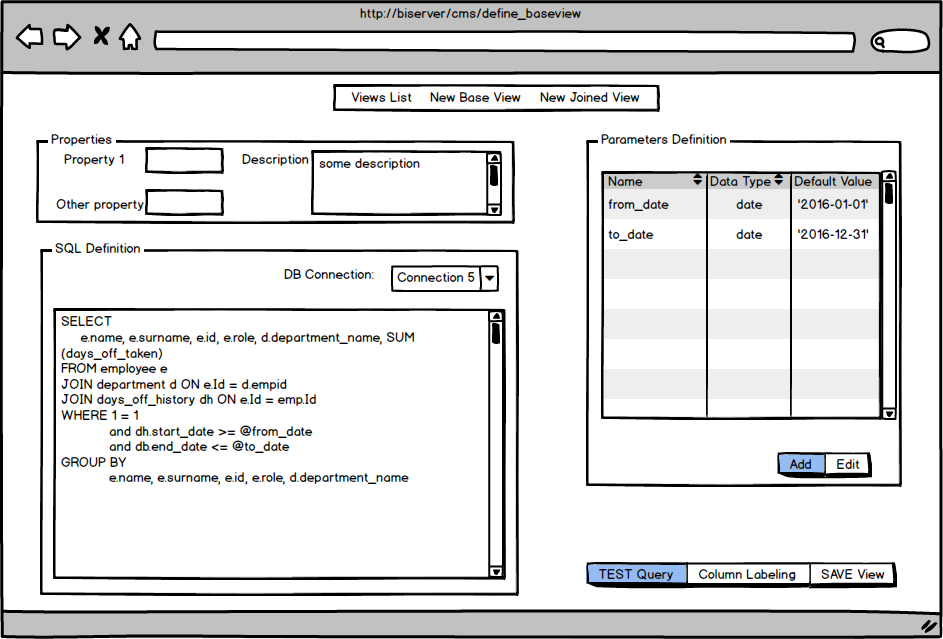
1. **System Management mockup**

Mockup A is an overview of the Central Management Console with a hint of system components to be managed. This CMS is intended to be used primarily by systems admins. Such a CMS needs to be a pragmatic and clean GUI, providing a gateway to managing other components such as Database Connections repository or User Creation and Permissions.

1. **User Setup**

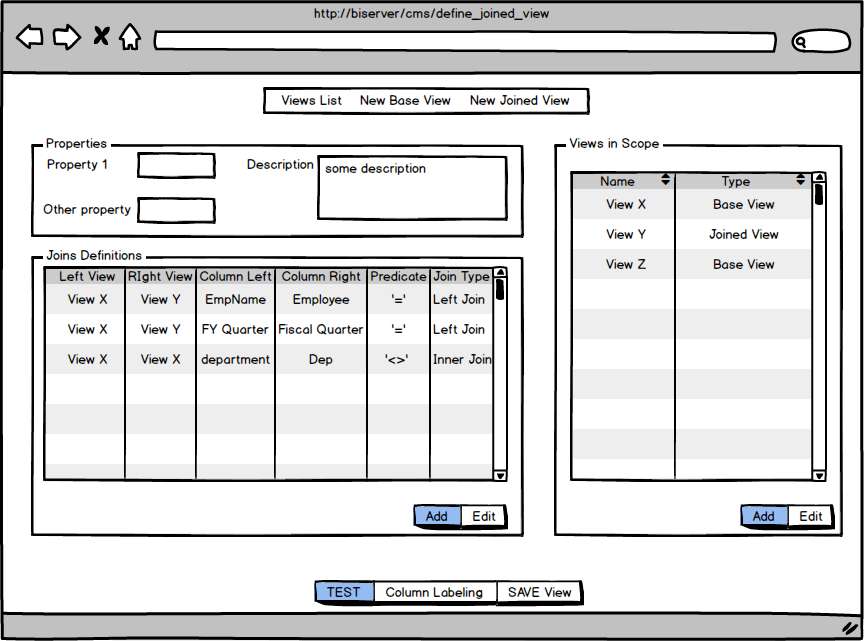
The User Setup GUI is an example of system management component accessed through the CMS. A system admin would use this interface to setup and assign a user to a permissions group, or even grant extended custom permissions that fall out of predefined setups. Also, this mockup gives a hint of the actual workflow that a user must follow in order to be granted with access to the system, i.e. a manager needs to provide an approval for any access requests.

1. **Develop SQL Views**

****

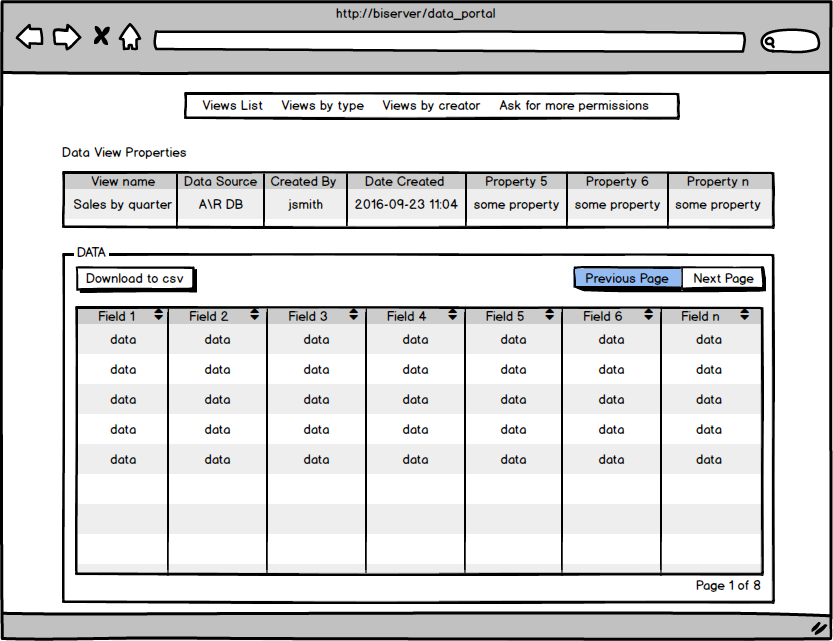
Mockup C represent the GUI that IT Developers need in order to develop Base SQL vies. As can be seen, the list of items in the menu bar is different than in the two previous UIs, suggesting clearly that this interface is destined to a different type of users. The GUI provides a section for View properties to be setup, and also the main components for defining a SQL query and necessary parameters. An SQL developer has the option to choose what connection his View will target from the available connections repository defined by the Sys Admin and which are available for him to access based on permission levels. From this screen, the developer user can choose to test his SQL queries but also to navigate to a secondary interface where he can further define column labeling.

1. **Develop Joined Views**



Mockup screen D is somewhat similar to the previous, and details how users will be able to create new Joined Views within the BI System, based on other pre-defined data views. The GUI offers the possibility of selecting Join Members from existing views list (Views in Scope group) and an interface in which to define the actual join rules. Joins between views will be possible by supplying the columns on which to be linked and also the predicate rules to be applied (in Joins Definition group).

1. **Consume data UI**

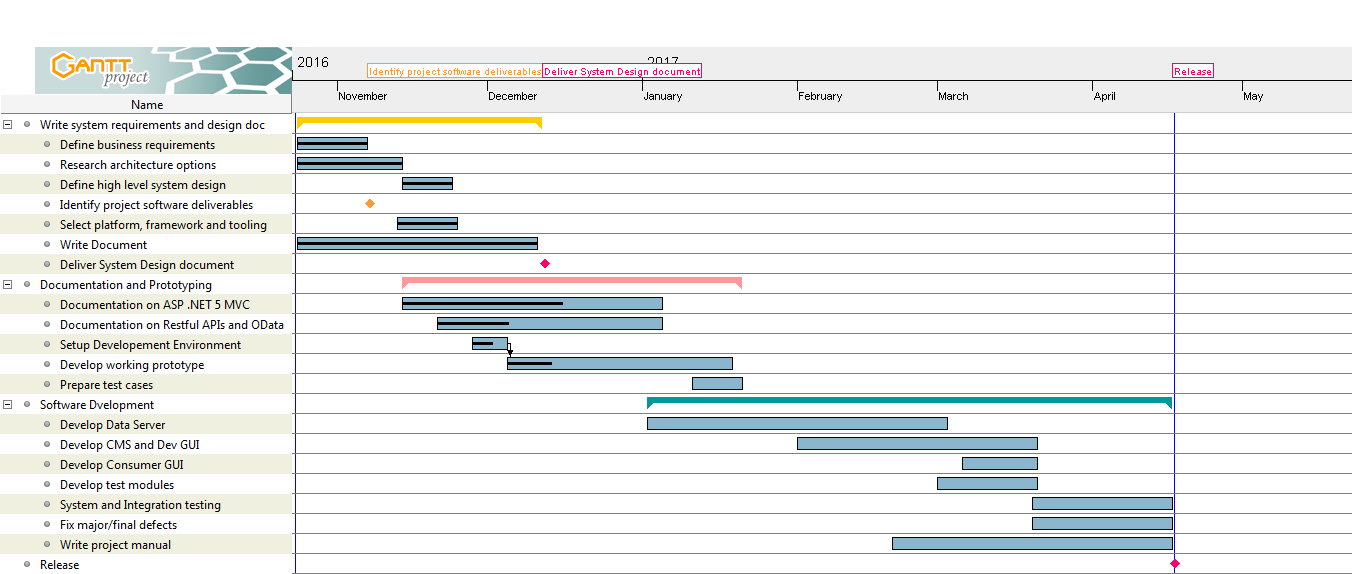


GUI mockup E presents the imbeded option of the BI System on how to consume the Data Views. Users, based on permissions will be able to enter this screen through a Data Views presentation list. From the GUI users can sort data and also have the option of downloading a CSV (comma separated values) file version of it for their own use. As mentioned in the document by now, the system must provide a WEB API too in order to make data available for consumtion directly through other tools, such as a spreadsheet application.

## 6.Project State, future work and risks

The effort of putting together the Systems Requirements and Design document proved to be an excellent method of also identifying the magnitude and work necessary for the whole project. The following paragraphs will summarize the state of the project and what work is left to be done until the final submission date.

Having identified the architectural components helped contouring the plan and milestones necessary to complete the project but also the dependencies and possible risks that might affect delivery. The following diagram represents the set of tasks and working time frames identified by the document author:



Tasks and work efforts are displayed in a Gantt chart, with project tasks on the left hand side and a visualization of the work effort times on the right, in the form of blue horizontal columns. As a short diagram legend, the black lines that cross the time frame columns represent the actual progress as of to date. The tasks are grouped into 3 phases:

* System Requirements and Design
* Documentation and Prototyping
* Software Development

As it can be seen from the diagram, at the date of submitting this document, the first part of the project has been completed and also the *Documentation and Prototyping* phase is ongoing. This second phase comprises mainly of documentation (research and studying) of the two frameworks selected for the project but also more technical parts like setting up development environments and in the end achieving a working prototype.

The documentation part is an essential to the success of the project as the author isn’t specialized in the .NET web platform (ASP.NET MVC) nor Web services development, so an important effort must be channeled to achieve a good learning curve and acquiring these skills. Furthermore, given this fact, the 2 related tasks (Documentation on ASP .NET 5 MVC and Documentation on Restful APIs and OData) could be ticked as being a risk. The development times and quality of the final software project are dependent on how well the documentation is being carried out.

Another two tasks ongoing at the date of submitting this document are *Setup Development Environments* and *Develop working prototype*. Software platforms and necessary IDE’s (.NET components, Visual Studio 2015, MS SQL Server Express, SSMS, Erwin) have been installed and setup on the development machine. Furthermore, 3 distinct database engine instances (MSSQL 2012, Oracle 11g, IBM DB 2 LUW 10) have been installed on a virtual machine (Virtual Box). The 3 databases will be necessary for development/test work as playing the role of target operational systems databases. Next step necessary to complete this project task is to prepare/generate sample data.

The *Develop working prototype* task presents high priority as when this is completed it can offer an even better picture of the rest of the work left. When developing the prototype, the developer (author in this case) can better estimate the development tasks from phase 3 and maybe readjust the plan. Also, during or after prototyping is completed, the developer could discover that an architectural approach or path may not be feasible so certain designs need to be reconsidered. This task itself carries an important risk if not completed in time, because if delayed it may be too late to reconsider any rework.

Regarding future work in the project, the 3rd phase comprises of software development and testing. The main chunk of work and most important task in the phase is *Develop Data Server*. As explained throughout this document, the Data Server is the main component in the system so completion of this task is highly dependent on the documentation and prototyping work. Moving towards of the project, the set of testing and bug fixing tasks are directly related to the quality of the end product.

1. By first hand I mean small issues which didn’t require engineering skills or more complex repairs that the manufacturer can offer on warranty. For example: changing parts of a mechanical fixes [↑](#footnote-ref-1)
2. Work Force Management System [↑](#footnote-ref-2)
3. Customer Relationship Management System [↑](#footnote-ref-3)
4. I.e. making sure that different systems would have the same definitions for customers, employee data, products and services. [↑](#footnote-ref-4)
5. Much like all SMEs in Ireland [↑](#footnote-ref-5)
6. https://en.wikipedia.org/wiki/PHP [↑](#footnote-ref-6)
7. https://en.wikipedia.org/wiki/Python\_(programming\_language) [↑](#footnote-ref-7)
8. https://en.wikipedia.org/wiki/Java\_(programming\_language) [↑](#footnote-ref-8)
9. https://en.wikipedia.org/wiki/.NET\_Framework [↑](#footnote-ref-9)