TÉCNICO LISBOA

Object-Oriented Database Edit and Analysis System

Nuno João Rodrigues Gomes

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Pedro Manuel Moreira Vaz Antunes de Sousa

Examination Committee

Chairperson: Prof. Nuno Miguel Carvalho dos Santos
Supervisor: Pedro Manuel Moreira Vaz Antunes de Sousa
Member of the Committee: Prof. Flávio Nuno Fernandes Martins

November 2022

Acknowledgments

I would like to thank my family for the financial and emotional support, encouragement and care over all these years for always being there for me, without them this project would not be possible.

I would also like to acknowledge my dissertation supervisor Professor Pedro Manuel Moreira Vaz Antunes de Sousa for their insight, support and sharing of knowledge that has made this Thesis possible. Thank the employees at Link Consulting who helped me and gave me their feedback.

My teachers during my childhood, for helping me understand how important it was for me to educate myself and to be educated by others.

Last but not least, to all my friends and colleagues who helped me grow as a person and who were always there for me during the good and bad times in my life. Thank you.

To each and every one of you – Thank you.

Abstract

Nowadays possessing data is essential for companies, scientist, governments and for that it is really important to have efficient databases to store that data. There are different types of databases, all have the same goal to store data, but with different methods and with it's advantages and disadvantages. In this thesis the main goal was to develop a system to edit and analyze an object-oriented database, to implement this system I used Excel with the help of Visual Basic for Applications (VBA) and some Python scripts to interact with the Application Program Interface (API) which therefore interacts with the database. The database is printed in the Excel sheets, every sheet is a different class, every row an object and every column a property of the object, the user can than edit the cells and update the database. It is also possible to create charts with the data from the database so the user can analyze the data. After testing I was able to realize that Excel is a very good program to edit a database due to its already implemented features to edit data simply and it's very intuitive tabular view of the data.

Keywords

Database Edition Tool; Object Edition;Import Database;Make Charts from Database.

Resumo

Atualmente, possuir dados é essencial para empresas, cientistas, governos e para isso é muito importante ter bases de dados eficientes para guardar esses dados. Há diferentes tipos de bases de dados, todos tem o mesmo objetivo de guardar dados, mas com métodos diferentes e com as suas vantagens e desvantagens. Nesta tese o objetivo principal era desenvolver um sistema para editar e analisar uma base de dados orientada a objetos, para implementar o sistema eu usei o Excel com a ajuda do Visual Basic for Applications (VBA) e alguns scripts de Python para interagir com a Application Program Interface (API) que por si interage com a base de dados. A base de dados é impressa nas folhas de Excel, cada folha representa uma classe diferente, cada linha um objeto e cada coluna uma propriedade do objeto, o utilizador pode editar as células e atualizar a base de dados. Também é possível criar gráficos com os dados da base de dados para o utilizador poder analisar os dados. Depois dos testes eu apercebi-me que o Excel é um programa muito bom para editar a base de dados por causa das suas capacidades já implementadas para editar dados de maneira simples e sua vista tabular dos dados muito intuitiva.

Palavras Chave

Ferramenta de Edição de Base de Dados; Edição de Objetos;Importar Base de Dados;Criar gráficos de uma Base de Dados.

Contents

List of Figures

List of Tables

Acronyms

API	Application Program Interface
JSON	JavaScript Object Notation
SQL	Structured Query Language
VBA	Visual Basic for Applications
OS	Operating System
XML	Extensible Markup Language

1

Introduction

Contents

With the exponential increase of the usage of technology in our lives, it has became more and more important to save data, data about us, data about our surroundings, data about other species, to store that data we use databases, the data is stored in tables with rows corresponding to the number of records of the table and columns with the properties we want to save for each record.

Having a system where it is possible to import a database and analyse the data, create charts, tables and even being able to edit the database in a tabular view is very useful for an easy and intuitive interaction with a database.

This thesis is about how to find a method to export, edit, analyze and then import again to the database with an object oriented interface, it is also about facilitating the creation of charts and other reporting information about a database, there are various types of databases, here there is more detailed information about it in the Types of databases section.

1.1 Context and Problem Description

Link Consulting, the company with which I did this thesis in collaboration, has a product called ATLAS, this product is used to help other organizations manage their projects and build an architecture for the organizations. ATLAS has a timeline that helps the organizations seeing how the projects will develop in the future, it has a time bar that can be moved to go in the past and future and see how the projects are predicted to evolve and how they were in the past compared to the present. ATLAS is also very good at collecting information from internal and external sources. [1]

ATLAS works with a database with an object-oriented interface, giving a small explanation, there are repositories that work as a database, every repository has classes, the class has properties defined by the creator of the class, it is possible to create objects that are associated with the class, and the user can then fill the properties with values to store. For a more detailed explanation of how their database meta-model works, it is explained in this section ATLAS database meta-model.

The ATLAS website is used to access the data in the repositories, every user has access to some repositories, the user can see and edit the data in the website, but it is not very practical because the user needs to edit object by object and class by class, so we decided to develop a system that helps editing the database in an easier and more practical way.

Another problem Link Consulting wanted to solve was to find a program where they could create charts, tables, or more statistics information to be able to make easier and more detailed reports about the data they keep, which is very important taking into account that ATLAS helps organizations manage their projects, so making reports with statistics about their strategies is really useful.

After researching about the already existing programs that could perform the tasks mentioned above. Together we arrived at the solution, which was to use Excel to print the database data and from there

be able to create charts, tables, and more statistics from the data. Using a simple interface that, as mentioned before, can also be used to update the database in a simple way, rather than needing to update object by object and class by class.

When developing the program, I had some expected problems during the implementation of the code, some of this problems will be explained in the Implementation Challenges section.

1.2 Goals

The objective of this work is to do a system with which we can have a intuitive, easy and efficient way of editing a tabular database, the other objective was to have a way of creating charts of the data in the database.

After unsuccessfully searching for already developed programs that would achieve this goals, we came to the conclusion that a good program to develop this system would be Excel because it already has a tabular view to see the data, it has a lot of developed features to create charts and more statistics of the data, and it is a very intuitive program to use. When I started working on the Excel system, we realized these goals would be applied:

- **Importing classes from a repository**: Every repository has multiple classes, the idea is to import each class into a different Excel sheet, and every row represents an object and every column a property of the objects.
- **Edit Data and then update the database**: To edit the data, the user can change the cells with the properties values and then have a button that, when pressed, would see what was changed and update the database.
- **Create charts to analyze the data**: Using Excel, it is really easy to create charts of data, using the Excel interface the user chooses the chart type and then selects the cells with the data to do the chart.

These goals were the initial ones, and of course, with the progress of the development of the system and with the feedback of the ones who used it, there was an upgrade of what the program does. The goals from the final result are more explained in the System Architecture.

2

Theoretical Background

Contents

2.1 Types of databases

”A database is a systematic collection of data. They support electronic storage and manipulation of data. Databases make data management easy.” [2]

Here are some of the different types of databases:

- **Hierarchical databases** - As the name indicates, the data is arranged in a hierarchical way, can be imagined as a parent-child relation, where the parent can have multiple children but the child just has one parent. It works like an IT tree, the top node can have multiple child nodes, but the child nodes can only have on parent. When representing in records like the common relational databases every record has its own ID, the properties you want to save, and the parent ID which points to their parent record. If a parent record is deleted, all it's child records will be also deleted. As advantage the database is really simple so it's really fast to iterate it, as disadvantage the database is really rigid, because for example if we want some child to have two parents, we need to create two child records representing the same record. [3]

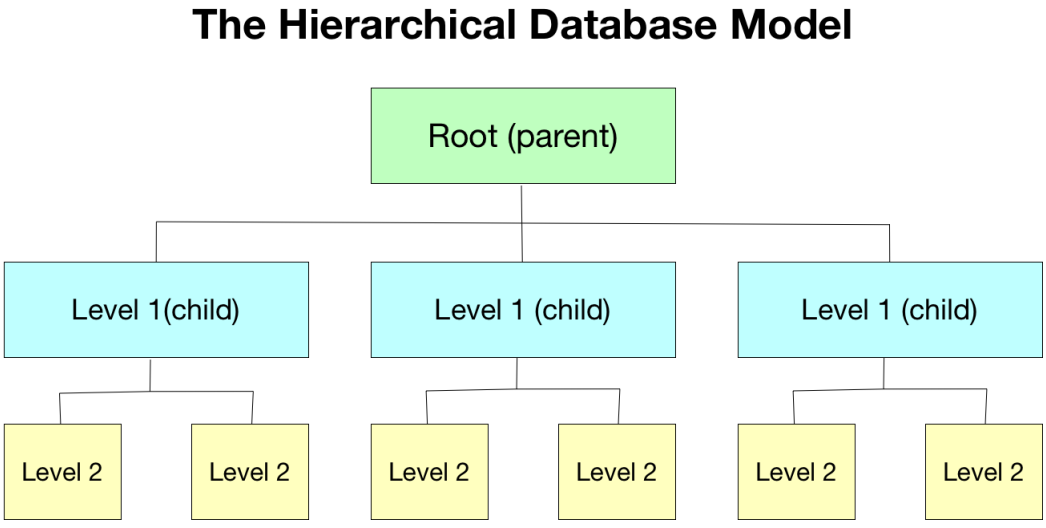

Figure 2.1: Hierarchical Database Model

- **Network databases** - It works the same way as the Hierarchical databases but the child can have more than one parent, it is better to represent two-directional relations. It supports many to many relationships which makes it easier to search in the database, giving it faster data access, flexibility and accessibility. As disadvantage it is difficult for first-time users and it is not good for maintainability, because it is really complicated to change the database model, any new information can alter the entire database. [4] [5]

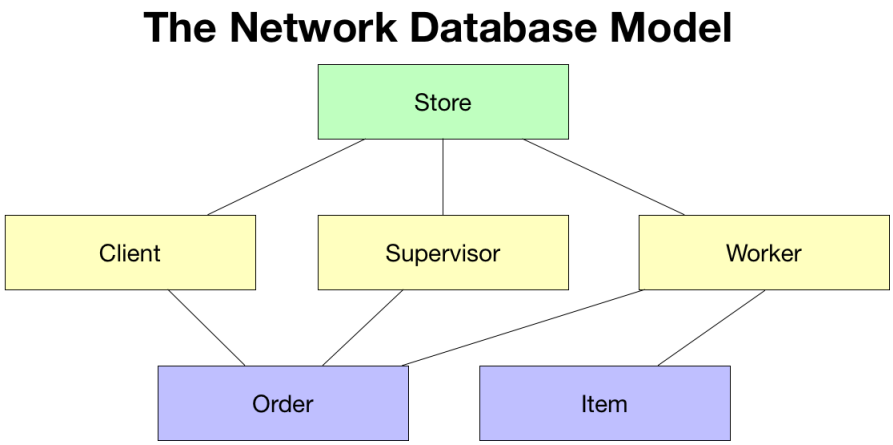

Figure 2.2: Network Database Model

- **Object-oriented databases** - A database that is made in object format, it is easier to call an object, it is possible to refer to other objects using references. The objects can be really complex with a lot of properties and methods, this model is excellent when using object oriented languages like Java, so when using objects we can store the whole object in the database and retrieve the next time we want it, to store an object in a relational database we would have to decompose the object to store all the properties as a table. As an advantage it makes the code much simpler and cleaner if using an object oriented language, there are no JOIN's like in a Structured Query Language (SQL) database so if it is necessary to do that, it would be more efficient. As disadvantage it is slower doing a simple lookup in the database, and there is not a universal language like SQL to store the data, every language might have it's own syntax for the database usage. [6]

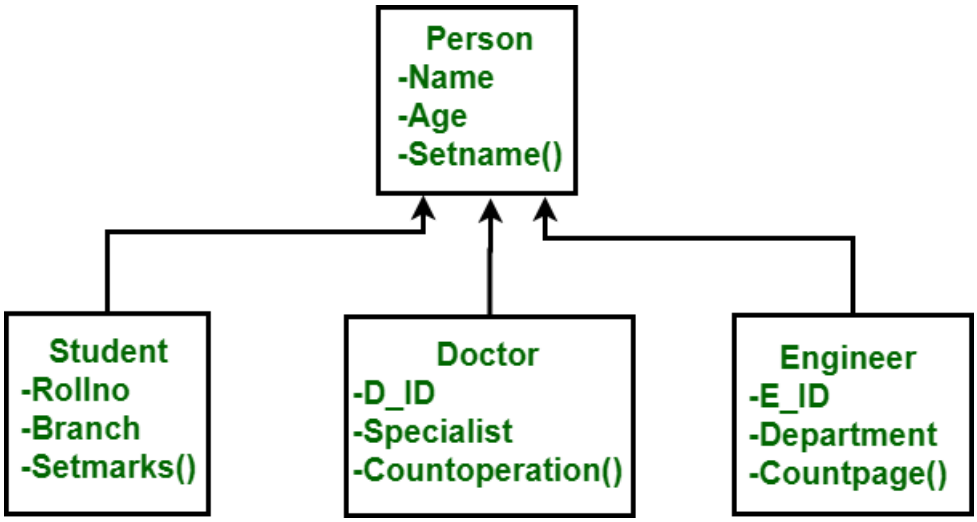

Figure 2.3: Object Oriented Database Model

- **Relational databases** - These are the most commonly used type of database, uses SQL as Application Program Interface (API) to communicate with the database, stores in tables, each row is an record with a key (unique identifier), each column is used to store a value. This makes it easy to identify relations between the data, the relations between the tables is made using the keys to identify the records. As advantage it is really good at categorizing data, you can modify the database without having to restart everything or changing any application, easy to use for having a common language to interact with the database, it has security because you can limit who can access the database and it has multiple access to the database from different users at the same time. As disadvantages, everything needs to be very well planned out, when doing the database, it is required to have developers maintaining and optimizing the database, is inflexible because if you want to add new data, you need to update the scheme model of the database and is not very scalable because it does not have very good performance using multiple servers. [7]

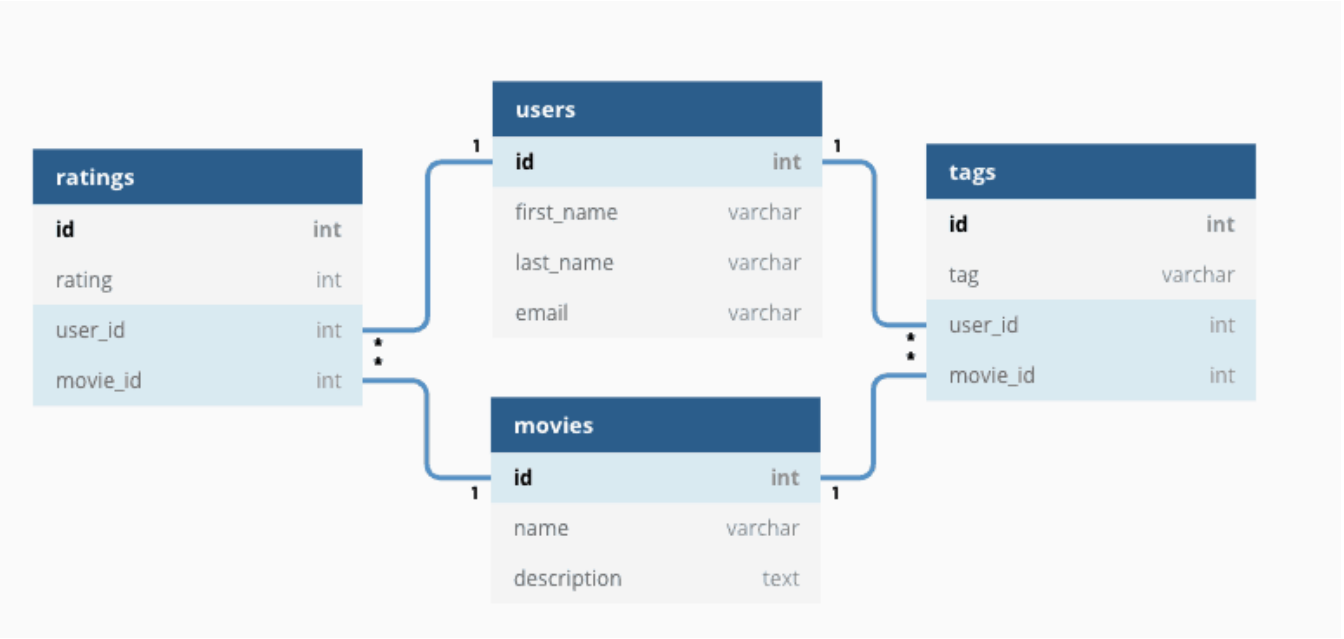

Figure 2.4: Relational Database Model

- **NoSQL databases** - A NoSQL database does not have a rigid schema, which makes it easy to scale and, therefore, good for large amounts of data. It allows for horizontal scaling, which is much more affordable than vertical scaling that is needed in relational databases. There are four different types of NoSQL databases:

  - **Key-value Pair Based**: It is the most basic model, it saves a hash table where there is a key that can correspond to a JavaScript Object Notation (JSON), blob, string, integer, etc. It is used to avoid having to make a scheme of the data.

– **Column-oriented Graph**: It works with separate columns, so every column is treated separately, so it delivers high performance on aggregation queries like ”SUM”, ”COUNT”, ”AVG”, ”MIN”, etc.
– **Graphs based**: It is a graph where every node is an entity and every edge is a relation between nodes, a graph database is a multi-relational which makes traversing through their relations very quick
– **Document-oriented**: Instead of using tables like in relational databases, it uses JSON or Extensible Markup Language (XML) to store the data, as there are no prepared properties to write on JSON each object can have different properties, making it very flexible.

As advantages it permits using Big Data, it allows horizontal scaling with fast performance, can handle data that are structured and also data that are not structured, do not need a high-performance server just for the database, simple to implement, and very flexible (easy to save different data). As disadvantages, there are no standardized rules to develop, not good with relational data, has a learning curve for new developers, and does not have consistency (having multiple transactions performed simultaneously). [8]

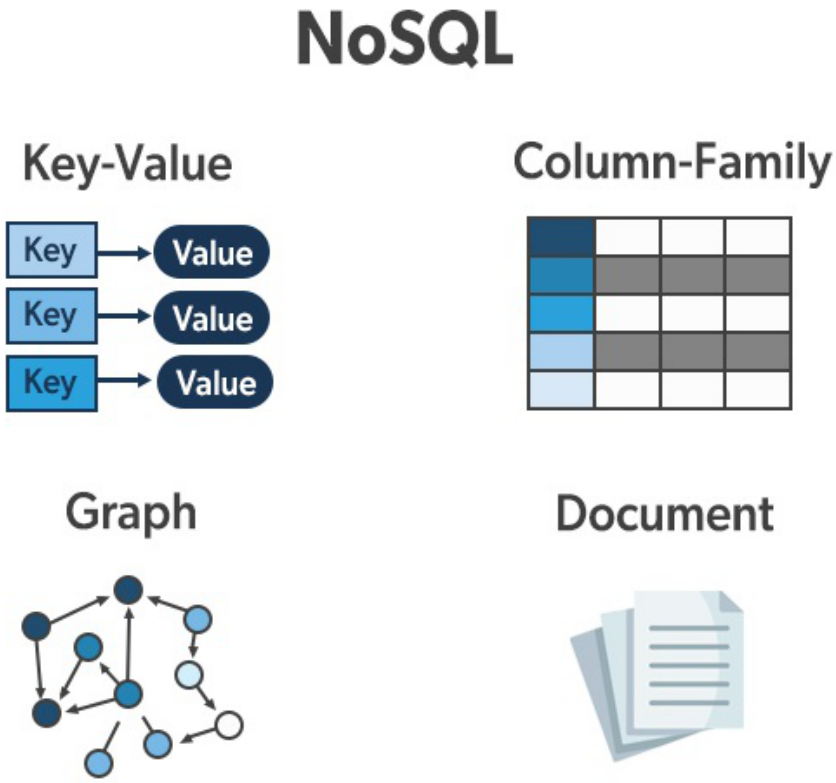

Figure 2.5: NoSQL Database Model

2.2 AppleScript

”AppleScript is a scripting language created by Apple. It allows users to directly control scriptable Macintosh applications, as well as parts of macOS itself. You can create scripts—sets of written instructions—to automate repetitive tasks, combine features from multiple scriptable applications, and create complex workflows.” [9]

In the Solution Description I will mention how the system uses AppleScript in the Mac Operating System (OS) version of the solution, so the program can call terminal commands, in our case it is needed to call python scripts and it is also used to create an alias of the Excel file in the installer script.

AppleScript is a scripting language like, for example, shell scripting which is used in UNIX Operating Systems, but is used to interact with the Mac OS features, here is a list of some of the capabilities of AppleScript:

- Tells applications to run certain commands, the commands are very diverse like change configurations, do tasks, etc...
- Interact with folders, files, for example: create folder, edit names of a file, move files, etc...
- Create an alias of a file
- Editing System Preferences, like change screen resolution, change desktop background, change dock setting, etc...
- Call commands in terminal and retrieving the result of the command

AppleScript has a lot of more features, it can do almost everything an user manually can, it is really useful to automate tasks and that is the reason I choose to use it.

3

Related Work

Contents

After researching about solutions that are close to mine, I realized that there was really not much made in this regard. I could not find any tool or prototype to edit an object-oriented database and not even much about visualizing and analyzing. I was able to find some tools to do what we need, but only for relational databases which are the most common ones, these tools could not be used because the ATLAS meta-model does not work with a relational database, and for that reason it was necessary to develop an independent solution without these tools.

3.1 Tools for Database Editing

On the Internet, it is possible find some examples of programs that allow an user to manage a relational database, create SQL queries, create tables, add variables to the tables. This is very common, there are a large number of applications that do it.

Then there are some programs that will let you, as an addition to those features mentioned just before, edit the database inline, so the program will open a window with the data of the database tables and will let you edit directly in the lines.

Here are some examples of programs that will let you do this:

- DbVisualizer [10]
- SQLGate [11]
- Postico [12]

These tools are used for relational databases, so they would not be able to solve the problem because the goal is to be able to edit and visualize a database with an Object-Oriented interface, which would not work with this programs.

3.2 Importing data into Excel

Excel has features that let the user import relational databases into Excel, it is a very good feature for data analysis, to create custom tables, charts and do statistics about the database data.

In the Excel tab Data, there is an option called Get Data, which then has inside an option called Get Data from SQL Server, then you need to insert the credentials to connect to the database and, of course, log in the database to authenticate, then there are some different options like importing a table or selecting just some columns from the table or running a SQL query to get some custom data.

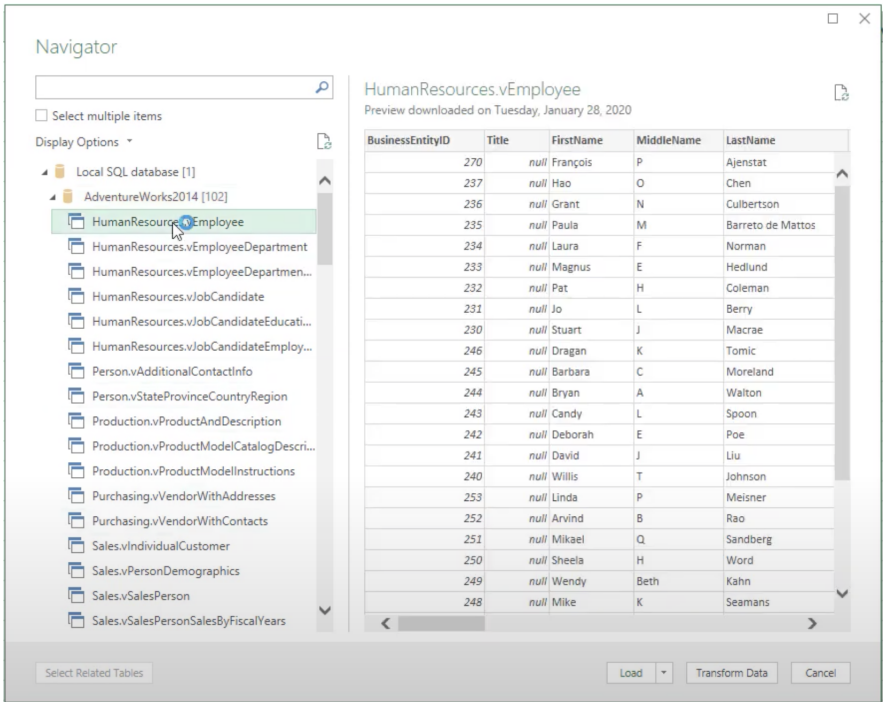

Figure 3.1: Database Import Popup

In the Transform Data button that appears in Figure Database Import Popup is where we can choose in a more custom way what to import in an interactive way using Power Query, it let you choose which columns to import and it even has a very interesting feature called Column from Example, which lets you write what you want to import as an example for the first row, for example if you have a column for First Name and one for Last Name and you write in the example the first name and last name of the first row, it will detect that you want those columns and will get those two columns for all the rows. So Excel really has a lot of possibilities so we can import the data in the most useful way.

It is also possible to import data from a website, so if we have a website that has a table in it, for example, we can connect the website to the Excel and it will detect the table and import that data, you can also edit the data the same way before with the Transform Data button. When Excel has made a connection with a database or a website, it will then have the option to refresh, which when clicked will automatically get the data again in case it has been altered.

In the Windows OS there are also other options to import data, import from Azure, Microsoft Query, Microsoft Access, you can also import from files: Text files, CSV, JSON, PDF, and some other options. These options are only available on Windows OS, in Mac OS it is not possible to use them.

These functionalities are, of course, only used for importing data, the connection Excel makes with the database is read-only, so it cannot be used to update the database.

3.3 Creating charts from a database

There are some online tools that create charts from databases, again the tools are prepared to receive data from a relational database, with a tool called Superset, the user can connect to a database and then do queries to get the data that will be later used to create the charts. The user can choose the tables and columns to create a data set that can also be used to create charts. This tool has a big number of types of charts that is possible to create, from traditional charts to more complex ones, maps, 3D charts, etc... [13]

Again, this tool cannot be used in this solution because the database we want to analyze is not relational and does not use SQL.

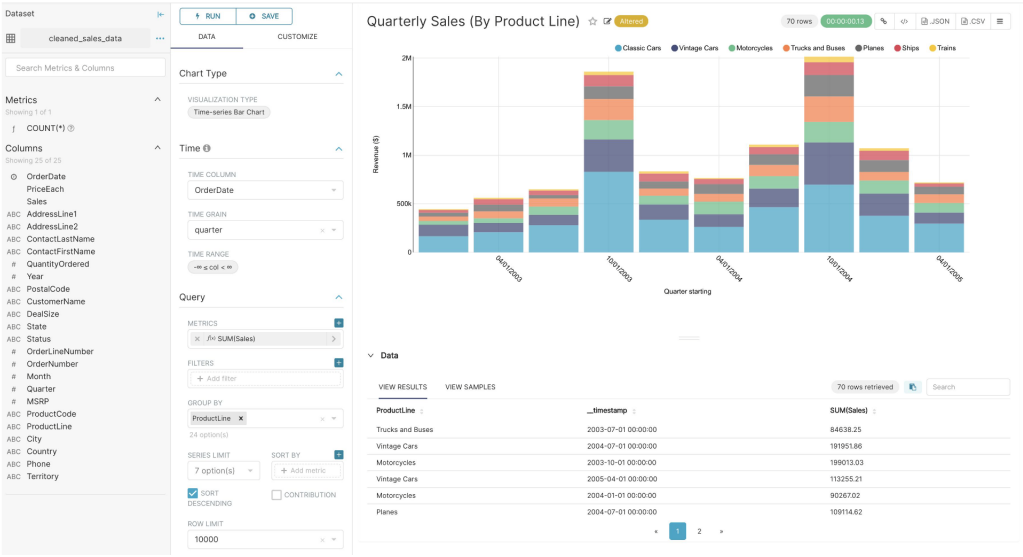

Figure 3.2: Example of creating a chart in Superset

4

Solution Description

Contents

Here I will explain how I did my solution, an explanation from the ATLAS database meta-model, a clarification about how I implemented it, an example of how it is used and give a little look at some issues I had during the development.

4.1 ATLAS database meta-model

In this section, I will explain how the metamodel for the ATLAS database works.

The database system works with repositories, every repository is like a small database, it has classes, in which every class must have one and only one repository, and then there are objects which are also from one and only one class. Then every object has properties to store information. The database works like an object-oriented database, it has objects and relations between the objects.

There are six different types of properties that the system accepts, those are (Numeric, Text, Boolean, Date, Hyperlink and Reference), that will of course change the way we need to write the values in the Excel and the way we need to update to the database.

The Reference properties are used to reference other objects, there is also an option to have a Class Restriction list, so that the reference can only be from object of classes that are in that list.

There are also profiles, profiles are like filters for the repositories, so if the user uses a profile in a repository, it will only present the classes that are filtered for that profile, instead of all the classes from that repository.

The ATLAS API is also ready when doing an update of the database to receive a parameter that is called update mode, it lets the user choose which type of update he wants, for example, just to do an update without erasing anything, or an update where it just adds new data, does not change anything that was already there. Here is a table with the modes that exist:

Table 4.1: Update modes

Update Mode	**Explanation**
Non Destructive - Additive	It only adds the new properties, it does not change properties that already have a value, this is the default mode.
Non Destructive - Update Only	It does not add new values to empty properties, only updates properties already with a value.
Destructive Update	It changes everything to be like in the requests we are doing, it overwrites everything.
New Data Only	It only writes new data, so in the Excel only let's you create new objects and properties.
Missing Data	It can only edit blank properties, it can also create new objects.
Merge update	It merges the information that is in the request with the information already in the database.
Delete	It deletes the objects that are referenced in a request with this mode, does not work for this system because it is not made to erase objects.

4.2 System Architecture

The system uses two programming languages (Visual Basic for Applications (VBA), Python), there is an Excel workbook, with a sheet called Control Panel, where there are inputs and buttons to do all the actions available. The VBA code is used to manage the Excel workbook, it is also in the VBA code that it is called the Python scripts. The Python code is then used to communicate with the ATLAS API. The API then interacts with the database.
The system allows you to do the following actions:

- Importing all classes and properties from a repository
- Importing some chosen classes and properties from a repository
- Edit Data and then update the database
- Create new objects
- Create new properties
- Create charts to analyze the data

I decided also to create an Installer, to make it easier for the users to install this system, the user chooses a folder to install and the script will copy the files and create a shortcut for the Excel file in the Desktop.

The system has an error display feature, which will tell in which sheet and cell the user is making mistakes when editing the data. So it is easy to understand what is wrong with what you are editing. It will be further explored later in this chapter.

I started by developing the Python import script, as it is the best way to get familiar with the API and the best way to know what I would be working with.

The import has 2 important parts, first we get information about the classes we are importing, and prepare a object list, from that information, we will then get the information for each object of each class and write in the object list. After obtaining all the necessary information, it's written all in an Excel sheet and in a JSON file, as we cannot write in a opened Excel file, we need to write in a extra one, and then within the VBA copy it to the one opened by the user. I added later the option to import just some classes and just some properties.

Then I decided to make the python script to update the database with the alterations the user had made on the Excel file, for efficiency reasons, instead of updating all cells, we will use the JSON file and the extra Excel to compare what are the differences between the edited one and the original one, after verifying if the type of variable is correct and every edition is correct than we do the updates.

In the VBA code is implemented the control over the Excel workbook, it is used to call the python scripts, clear the sheets if requested, create custom drop-downs depending on the data received, making sure there are no errors and if there are create popups to inform the user of the errors.

The most complex VBA script is the chart creation, it will automatically get all the values from a property in all the class's objects, and it will create a chart with just a Y axis or with also an X axis. More complex charts can be easily created using the intuitive user interface from Excel.

When the code was all finished and I started testing the code in a Windows computer, taking into account I developed the code in the Mac OS, I realized that I needed to make a version just for Windows because in Windows the calls to the Python scripts would have to be with a different method and the installer also had to be different because installing in a Mac OS requires different Python libraries than in Windows.

4.3 Example of Usage

4.3.1 Instalation

To start using the program, first the user needs to choose which operating system he is going to use the system in, to install the program it's used a Python script called install.py, it will ask the user for a directory to install the program, then check if it was already installed there and if so ask the user if it wants to repair the program by installing on top of the files. It will install the files, install the Python pip libraries and create a shortcut for easy access to the user.

4.3.2 Control Panel

Then when the user opens the shortcut file, this is the start Interface from the Control Panel sheet, there is one thing that is the most important and will only be written once and that is the path for your Python executable, the path the computer calls when you run a Python file. The second obligatory input is the repository name, it is needed to be able to import any repository, then there is the profile which is optional and works like a filter for classes, so instead of importing all classes, it can have a profile already prepared to just import a few chosen classes, then there is the update mode, which is a dropdown to choose the mode you want to import with, it is only used in the Update Database function. To import a repository, click on the Import All Repository button.

When you want to choose just a few classes then you need to click in the Get Class List button which will give you a list of classes to choose from. There is a column called Option with three options: *All Properties*, *Only These* and *Except These*, with *All Properties* it will import all properties of the class, with *Only These* it will import only the ones on the list to the right, and with *Except These* it will import

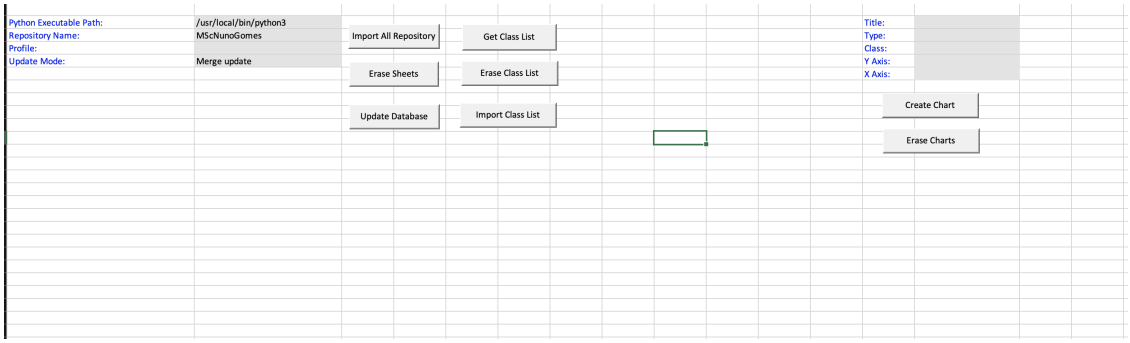

Figure 4.1: Control Panel Interface

all but the ones on the list to the right.

Class Name	Import	Option	Properties
Alarm	X	All Properties	
Application Component	X	All Properties	
Application Service	X	All Properties	
Assessment	X	All Properties	
Business Actor	X	All Properties	
Business Object	X	All Properties	
Business Process	X	All Properties	
Business Role	X	All Properties	
Classification - Communication Layer	X	All Properties	
Classification - Communication Paradigm	X	All Properties	
Deliverable	X	All Properties	
Goal	X	All Properties	
Location	X	All Properties	
Node	X	All Properties	
System Software	X	All Properties	

Figure 4.2: Class List Table

Then two drop-downs, one for the class and another for the property will appear and a button called Add Property will also appear to write the properties on the Properties column in the Class List Table. Here is the list:

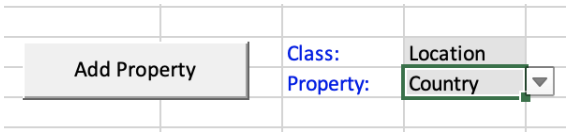

Figure 4.3: Add property to the table Interface

Showing now the result of an imported class, here is where you will be able to read, edit and analyse all the data imported. Every class will be in a different Excel sheet, the first row is the type of the property, the second row is the name of the property, and from there on, each row represents an object of that class, and each column represents a property from the class, with their values from the objects. The text in red font means that it is a reference.

Type: Text	Type: Text	Type: Text	Type: Numeric	Type: Hyperlink	Type: has goal Inverse Type: belongs to goal Class Restriction: Goal	Type: managed by Inverse Type: manages Class Restriction: Business Actor	Type: Text	Type: owned by Inverse Type: owns Class Restriction: Business Actor	Type: Date
Name	Country	Description	Employees Number	External Documentation	Has Goal	Manager	Observations	Owner	Productive Date
Bologna	Italy		10						
Frankfurt	Germany		21			Andreia Silva		Andreia Silva	
Leiria	Portugal		15				ijdoaidasodjs		
Lisboa	Portugal		60		Sell Online		Sede da Empresa		12/12/2022
Madrid	Spain		190						05/07/2023
Paris	France		130						
Porto	Portugal		30						
Roma	Italy		50						

Figure 4.4: Example of an imported class

4.3.3 Editing Data

As mentioned before, it is possible to edit the data, there are three types of edition the user can do: alter a value to a different value, add a value to an empty cell, and erasing the value of a filled cell, if all the types the user edited are correct, it will be updated in the database, after you click on the Update Database button on the Control Panel sheet. In the Figure 4.1 you can also see an Import Mode input that has a dropdown with a few different modes of updating the database, these are very important because if you use the wrong one you may end up erasing most of the data, you also need to check because some modes will only let you add new data, or just let you edit already filled data so it is really important to choose the right update mode.

4.3.4 Create New Objects

The system also lets the user create new objects, for that the user needs to add a value to the name column in the cell after the last filled row, it is only required to add a name property, but if wished, the user can already add more properties, and it will all be added.

4.3.5 Create New Properties

The user can create new properties too, for that he needs to add to the cell next to the last filled column in the sheet, it is required to add information to two cells, in the first row cell it is written the type of property, in the second one just needs the name he wants for the property, like it is done for the other properties, if it's one from the regular types of properties, he would just need to write "Type:Numeric" for example, if it's a reference you need to add a type with a name you wanted, it can be already created or a new one, then it needs an inverse type, and if desired you can also restrict the classes for this references using the Class Restriction.

4.3.6 Generating Charts

In the Control Panel, there is an option to easily create simple charts with one or two axis, the user can create charts by choosing which property from each class to use, and the system will automatically get the values and put them on the chart.

These are the types of charts the system can create:

- Bar
- Line
- Line with Markers
- Column
- Pie
- Area
- Doughnut
- Dispersion
- Dispersion with Soft Lines
- Dispersion with Soft Lines and Markers

If the user wants to create more complex charts with grouped data or with some more complex types such as 3D charts, it can be easily created using the features of Excel to create charts.

This is an example of input the user can use to do a chart, the Title is the title of the chart, the Type is the Type of chart, then chooses the Class he wants the properties from, after that two dropdowns will appear with the properties of the classes so you can choose which values to use.

Title:	Test Chart
Type:	Line with Markers
Class:	Location
Y Axis:	Employees Number
X Axis:	Name

Figure 4.5: Example of input for a chart

This would be the result of a chart with these inputs, this chart was created from the data that is shown in the Figure 4.4, with Employees Number as the Y Axis and Name as the X Axis, it then automatically fills the legend for the Axis and for the title.

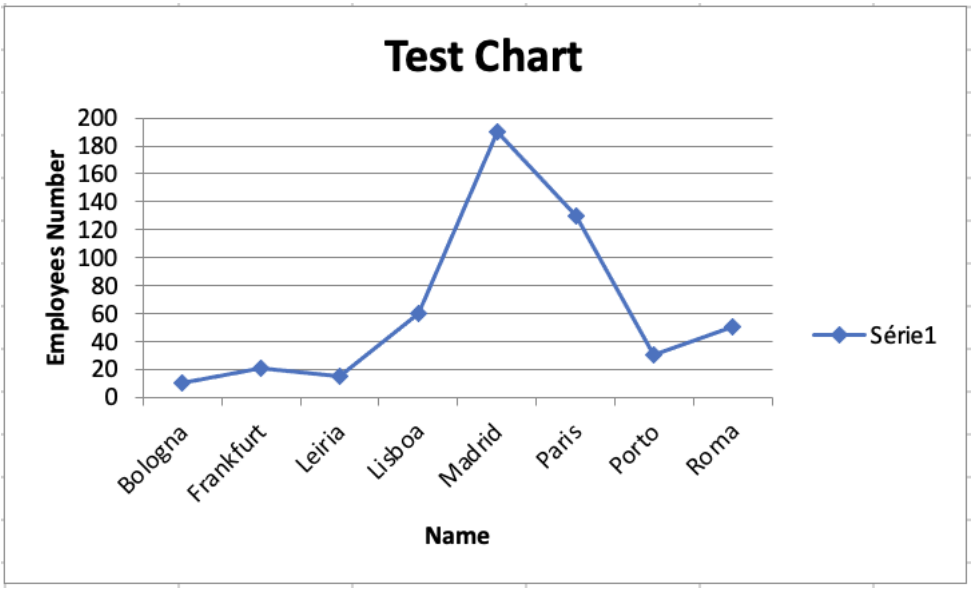

Figure 4.6: Example of a chart

4.4 Implementation

Here, I will explain paragraph by paragraph what each module and Python script does and I will also present some diagrams with a general explanation of the system.

Starting with the installer, there are two versions of the installer, one for Windows and another for Mac OS, but in reality it ends up doing exactly the same, just with different Python libraries. The installer is a Python script called install.py that starts by asking for a directory to install the program, then it checks if this directory was already used and if so asks if you want to repair the program, by installing on top of what you already had, it will then copy the files to the designed folder and it installs the pip libraries needed to run the program, in the Mac OS it needs to copy a AppleScript file that will then also be called by the VBA, to end it will create a shortcut/alias in the Desktop for easy access to the program.

The VBA Module1 corresponding to the Import All Repository button starts by getting the inputs for the Python script from the cells that the user is expected to edit, then it is called the login pop-up asking for the username and password to authenticate in ATLAS API, it will store the login credentials, so the user does not need to always insert the login data. After erasing any extra sheet if they exist, it calls the importRepository.py script (when calling the Python scripts the code will differ in the Mac OS and Windows version, in the Windows version you just have a method you can call to run commands on the console, in Mac OS you need to call an AppleScript which will then receive your terminal commands and run them), the script will get information from the ATLAS API depending on which parameters are sent to the function. In this module, it will call to import all the repository so it will get all the classes and properties from that repository. After the Python script ends, the VBA code will copy the workbook

created by the Python to the actual workbook that the user is now using. To finish this module it updates the dropdown used in the Chart creating inputs, it will be further explained in the correspondent module.

The importRepository.py script is used to get information from the ATLAS API to the Excel system, it is prepared to import all classes from a repository or just getting some classes and properties, depending on the inputs the script receives. It has 3 Python classes to store information from the ATLAS API, the Class, Object and Property. The script starts by organizing in a array which classes and properties to get in case the user just wants some classes and not all of them. The backup Excel workbook will then be configured to be written, and then the script starts calling the API to obtain the objects of the repository. It verifies which type of variable the object is and print it in Excel using the function of the pip library xlsxwriter for that type of variable, at the same time we save all the data in the Python classes I mentioned earlier. After iterating over every object we set the size of every column depending on the size of the string written there and it's created a JSON file that stores the information of the Python classes.

The VBA Module2 corresponding to the Erase Sheets button just erases all sheets of the Excel file except the Control Panel sheet.

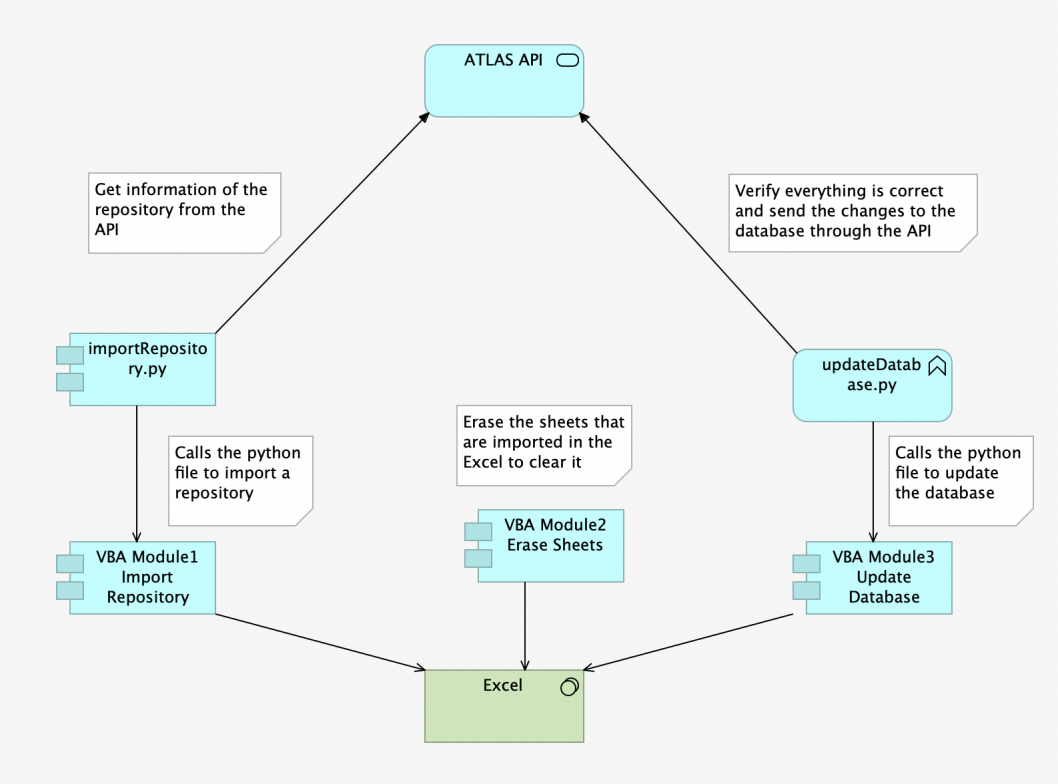

Figure 4.7: Implementation Diagram (1/4)

The VBA Module3 corresponding to the Update Database button, it gets the input from the Control Panel cells, the python executable path and the update mode, it will do the login popup if the Excel does

not have the login data already stored, it will then call the updateDatabase.py script which will detect the alterations to the Excel and update the database with them, after running the script it will read one of two outputs from the Python script, one means everything updated without errors, the other means it updated but one or more requests went wrong.

The updateDatabase.py script starts by creating a request queue, which is a class that stores every request for the API to do at the end of the script, if there are no errors in the user alterations, it iterates the backup workbook generated in the importRepository.py script and compares with the sheets the user altered, when it detects any cell different, it verifies if the type of variable the user inserted is correct and does a lot more verifications if it is a reference, all the errors that can happen are described in the Error Display section, if there is no error, it will add the request to the queue. This is for the easy cases where just some property is edited, then there are some cases where we need to do a lot of verifications, for example, if the user creates a new property, it needs to check if the types and everything correlates. In the end, if there are no errors, it will run all the API requests and return an output if there is no problem with the requests, and another if some request does not work.

The VBA Module4 corresponding to the Get Class List button, the module gets the same inputs as the VBA Module1 and will call the getClasses.py script which will get the name of the classes and the name of the properties, after running it copies the table with the class names from another workbook created by the script and prepares the dropdown by getting the property list from the script to add the properties to the property column in the class list table.

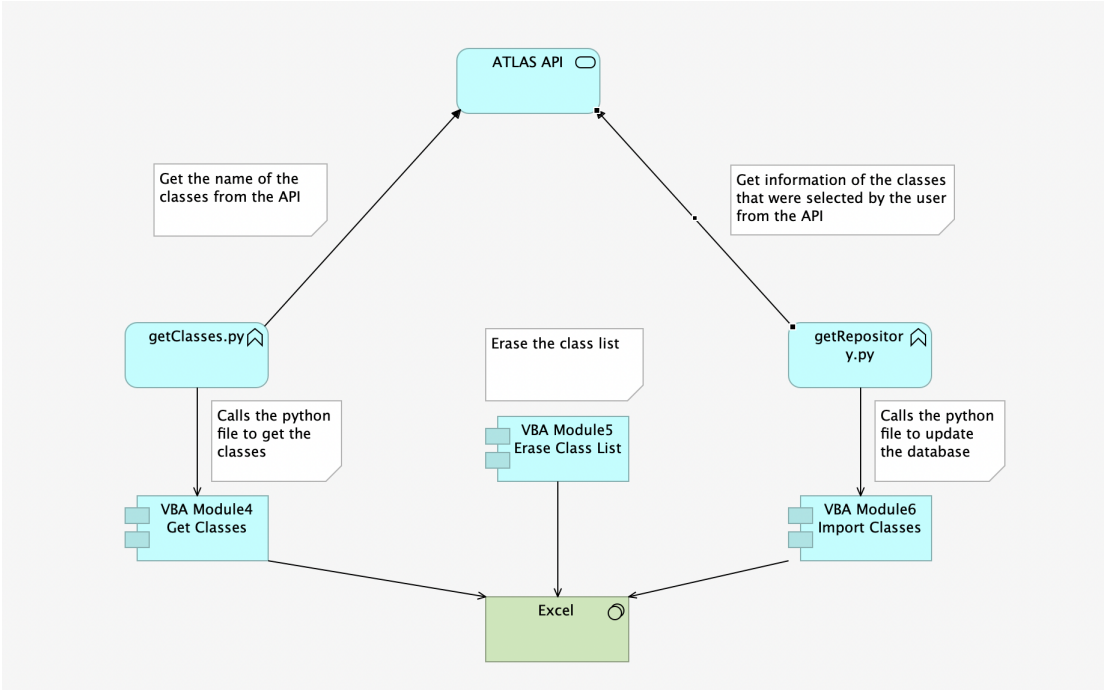

Figure 4.8: Implementation Diagram (2/4)

The getClasses.py script is used to prepare the class list table, which will then be copied by the VBA code, starts by getting information about all classes, and writes the table in a workbook, then writes also a list separated by commas of all properties of each class, which will then be used in the dropdown to add the properties to the column in the class list table.

The VBA Module5 corresponding to the Erase Class List button, it is used to erase the class list table, and it also erases the dropdown to add properties.

The VBA Module6 corresponding to the Import Class List button, it imports the information from the repository but only the classes and properties that were selected by the user, the code is almost equal to the VBA Module1, the only difference is that when it calls the getRepository.py script, it's sent a parameter informing that it needs to only get some classes. Then the getRepository.py script will iterate the Excel cells with the class list table in the Control Panel sheet and will store in an array the classes and properties to obtain.

The VBA Module7 corresponding to the Create Chart button is the most complex VBA module, it reads the info from the Chart inputs and immediately gives an error if any is not filled. Then it will get the address of the cells the user selected to print in the graph, by comparing the name of the columns with the property you chose. After that it creates a sheet called Charts to print the chart, then depending on the type of chart chosen, it will have different code, when the chart is all configured with the name, name of axis and the data inserted, it runs a mathematical function to calculate the coordinates of the chart so it prints in grids of 2 charts by row.

The VBA Module8 corresponding to the Erase Charts button simply erases the sheet called Charts where the charts are printed.

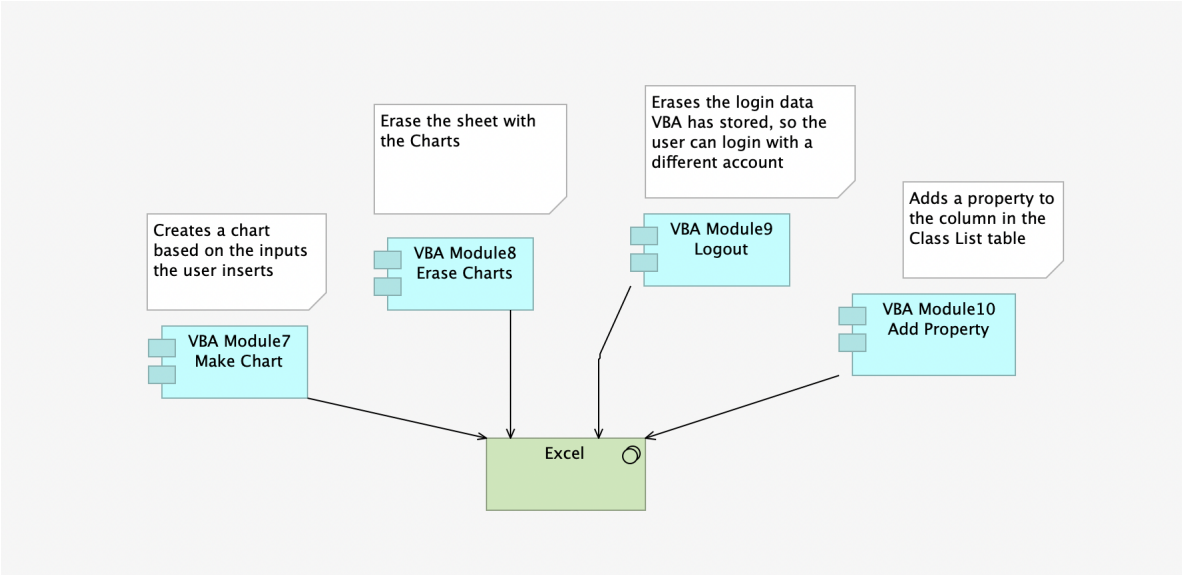

Figure 4.9: Implementation Diagram (3/4)

The VBA Module9 corresponding to the Logout button, erases the login data that VBA stored when the user used the login pop-ups, it is helpful if the user wants to logout and login with a different account, the Logout button only appears on the Control Panel when you have already logged in.

The VBA Module10 corresponding to the Add Property button, it reads the Class and Property inputs and inserts the property chosen in the Properties column in the Class List table for the chosen class. If there's none there it just writes it, if there is already one, it adds with a comma. The inputs and the Add Property button appear only when the Class List table is called in the Get Class List button.

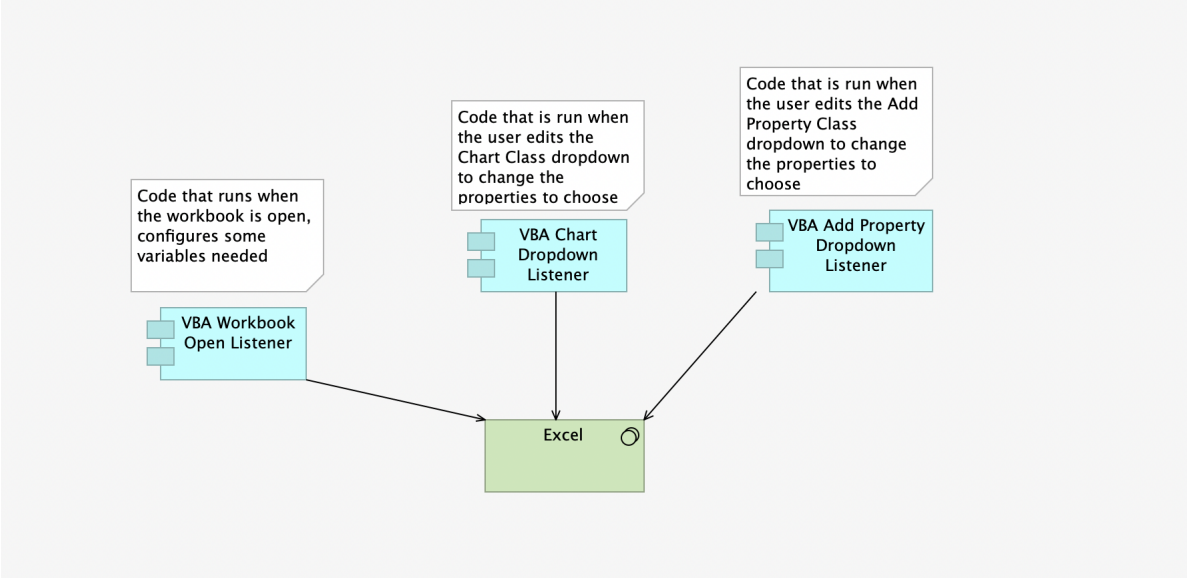

Figure 4.10: Implementation Diagram (4/4)

There is also a code that runs when the user opens the workbook, it works like a listener that runs every time the workbook is opened. The code prepares the Control Panel sheet for the utilization of the user, it prepares the class and type dropdown in the Create a Chart section, and it adds the values to the update mode dropdown, it also checks if the Class List table is present to get the Add Property dropdowns.

To end, there are also two listeners that are expecting the user to change the Classes dropdowns, to then change the Properties dropdowns. It is used in the Chart inputs and in the Add Property dropdown, but with different methods, the chart listener will get the properties cell by cell existing in the sheet of the corresponding class, the Add Property dropdown, it saves an array with all the properties when we get the Class List table, so it just needs to assign the correct element of the array to the dropdown.

Color Legend:

- VBA Modules
- Button Names

- Python Scripts

4.5 Error Display

As we are using an application like Excel, it will give a lot of freedom to the user, that means that there are a lot of space for errors that can happen.

There are two types of error displayed, the errors that will just give a simple error message, and errors that will have information about which cell of which sheet is being incorrectly used. The system will also print the error if there is an exception in the Python scripts.

There is also the fact, that this is not a system that is open to any user and is intended only to be used by Link Consulting employees. That means that the users that are using the system already have a lot of knowledge about it and after reading the User Manual it makes it even easier to use.

A table with a list of the errors that can happen and their explanation is presented in the appendix, Error Display Table.

In the picture below, we can see an example of an error, when an user has edited the data and tried to update the database, but he had an error in his edition, in this case it means that in the sheet/class Location in the cell D8 it is expected a numeric value and the user inserted something else.

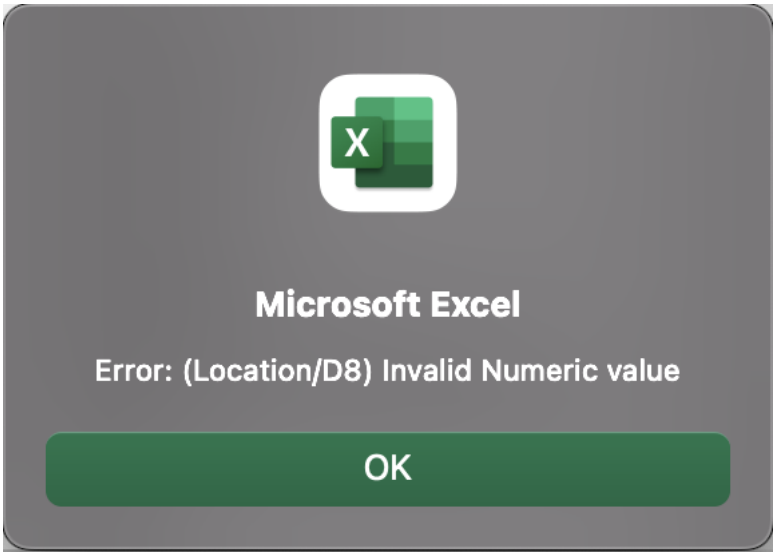

Figure 4.11: Example of an error

4.6 Implementation Challenges

The implementation issues started when we decided we were going to use Excel, because VBA would have some missing functionalities we would need, for example it would not be possible to call the ATLAS API, which is fundamental for the system to be able to import data from the database and to change the data on the database also.

To solve this issue, I had to create Python scripts that make the interaction with ATLAS API and then call these scripts in the VBA code.

In Windows operating system, it is really easy to call scripts with VBA, but as I was developing with a Macintosh I quickly discover that to call a script in VBA using the Mac OS I would need to use AppleScript, to be able to run the Python scripts.

Those were the main issues in terms of development, but then of course there were challenges because it was my first time using VBA and it is not a very intuitive programming language.

Another issue I would definitely consider difficult, was to control what the user can do in the Excel sheets, as Excel is such an open program, it let us do so many different things, I had to be really careful with what the user could do that would crash the program.

4.6.1 Concurrency Problems

The solution does have some concurrency problems, concurrency means when two or more computers are using an IT system at the same time, which by itself does not need to be a problem, nowadays almost every system needs to be ready to receive multiple concurrent requests.

This is the main cause of problems that can happen when updating the database, so if a user imported a repository and while editing someone changed something in the repository, once the user is going to update the database, the update is going to be made based on how the repository was when he imported it, in many cases there might be no problem from that, but next I will show some concurrency problems that can happen in this case, how the solution reacts, and what could be improved:

- **Creating an object with a name that was used by a concurrent user**, if someone creates an object with the same name we later created, then the system will give us an error, but nothing unpredictable will happen to the database, simply that requests will not work and will report an error to the user.

- **Editing an object that was erased by a concurrent user**, if an concurrent user erases an object you are editing, the system will give an error because it is trying to edit an object that does not exist and the alterations made will not occur because the object was deleted.

- **Editing a property that was erased by a concurrent user**, if an user deletes a property that you are editing a value or adding a new value, the system will work normally and will not give an error, but the values of that property in every object will all be erased.
- **Using a reference to an object that was erased by a concurrent user**, if someone erases an object that you referenced, the system will create a new object with the name of the object that got erased, of course the reference will work, but if the object had values in the properties that were useful for the user, unfortunately all the values would be lost.

No solution for this problem ended up being implemented, but now I will suggest a couple of solutions that could be done to solve this problem:

- One solution, that could solve this problem but would require an entire makeover of how the updates to the database were made would be having a listener that every time the user edits a cell it would automatically update the database, it would only need to verify if that property was edited before the edition and if so give a warning to the user and if not update the data, this is the solution that would be better to solve this problem in my opinion but would not be efficient at all, because every time VBA calls a Python script, which would be needed, Excel does not let you edit until the script is run. So it would be good to ensure everything is well updated without issues, but the performance would be so bad that it would be practically unusable.
- Another solution would be to have a locking mechanism where there could be a button, for example, to lock the repository or even just to lock some classes of the repository and while the user was editing the data no other user would be able to do so, which would solve this problem, but I also think it is not a very efficient solution because there is no need to stop other people from editing the data at the same time, when there could be just one or two or even zero conflicts in the edited data.
- The best solution in my opinion would be to do the edition as it is now, so the user edits what they desire and then click on the update button, after checking if there are no errors and before actually updating, using the logs from the ATLAS API updates to check if there are any conflicts between what the user edited and some differences that might have happened. I think this is the most balanced solution because it does not lock the database so everyone can still edit and in terms of performance even though it would be slightly worse than it is now, it would not make a big difference and would still be very usable.

5

Evaluation

Contents

Here I will evaluate my solution, first I will be comparing it in terms of editing the database with the actual ATLAS web interface, to see which one is more practical and efficient, do an analysis of the pros and cons of using Excel to do this program and show some performance tests.

5.1 Comparing with the ATLAS web interface for editing

In this section, I will compare what is better to edit the ATLAS database, the ATLAS web application, or my solution, I will check in terms of usability and efficiency.

5.1.1 Usability

To visualize the database objects, in the web application, you choose a repository and have access to the list of classes of that repository and when you click on a class, a list of objects for that class appears.

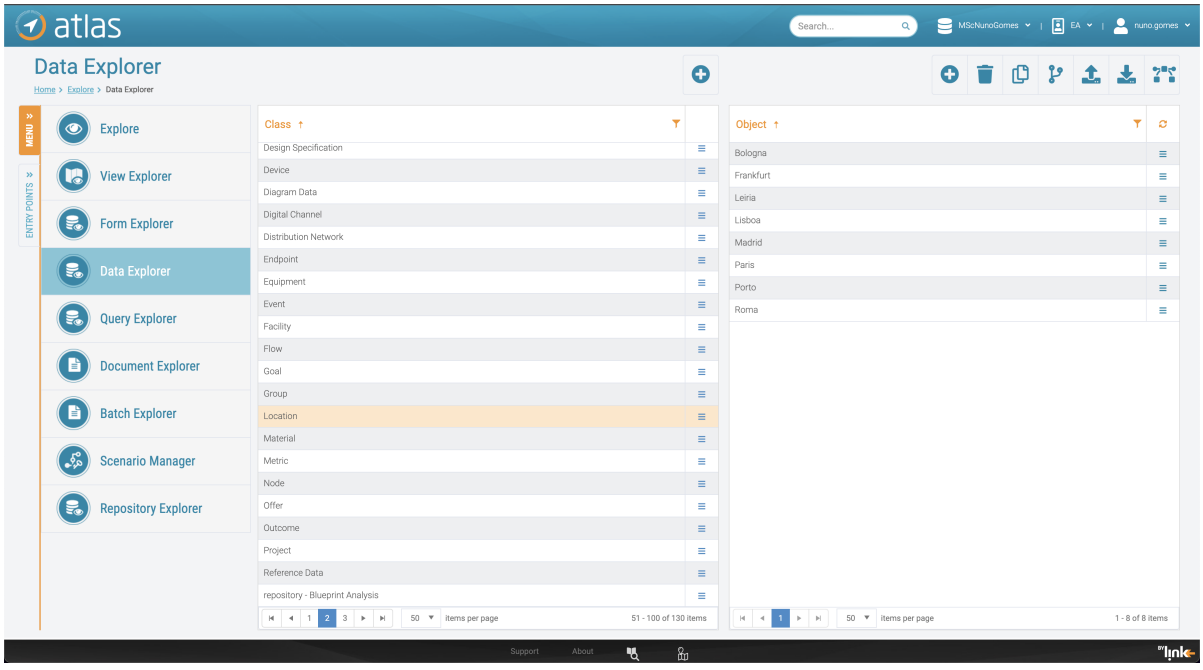

Figure 5.1: Atlas Web App: List of classes with the list of Objects

Then if the user wants to see the properties of an object, needs to click on an object to open a popup which will then let him see the properties of the values, sometimes the properties are divided in categories so the user needs to change between the categories to access all the properties.

To edit the properties on the web app the user needs to click on the pen icon on the right of the properties, then a input box will appear to insert the value, then after editing it is necessary to click in a confirmation icon, and after editing all the properties it is still needed to click on the Save button for the alterations to be made.

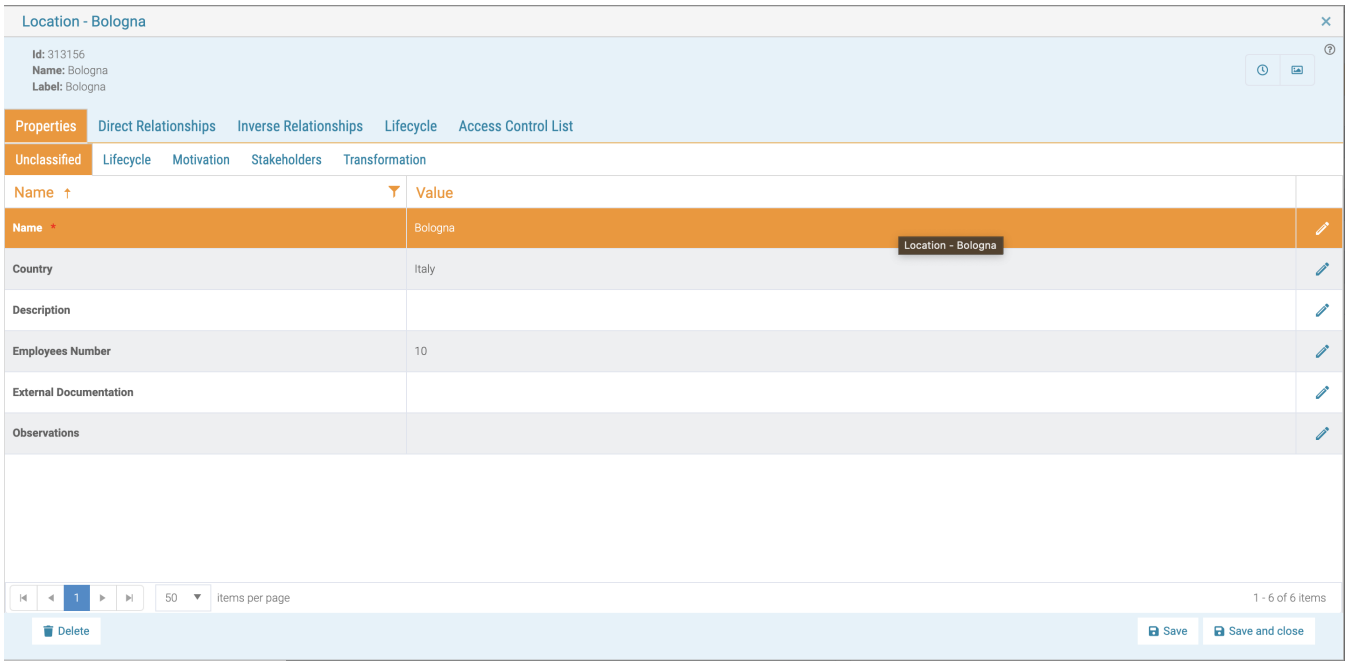

Figure 5.2: Atlas Web App: An object opened with a list of the properties

The ATLAS web app also has a tabular view, in every class the user can open a tabular view, he needs to click in the three tabs of the class, and then a couple of options will appear, one of them being the tabular view. Once the tabular view is open, the user will be able to see a table with only the name of the objects, if the user then wants to see the properties, needs to choose to add a property to see using the plus button. So if the user wants to have a global view of the class he would have to choose all the properties one by one, which is not very pratical, and when he wants to change from one class to another he would have to go back to the page in the Figure Atlas Web App: List of classes with the list of Objects and do the same for another class.

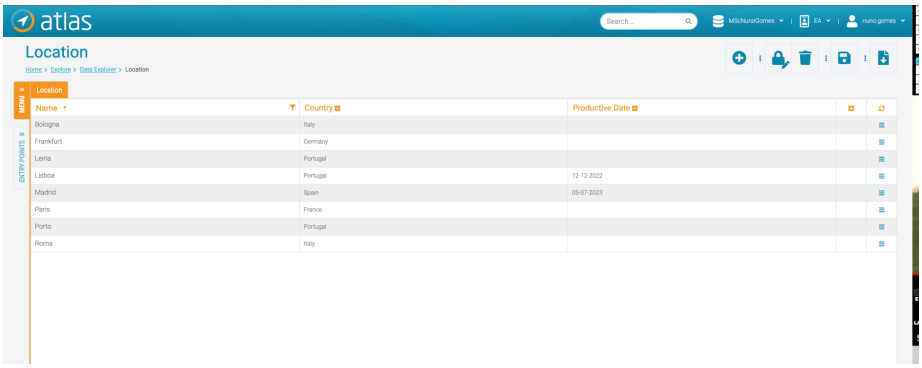

Figure 5.3: Atlas Web App: Tabular view

In my solution the user only needs to import the repository, after a few moments of the importing loading, every class will have a sheet, in every sheet there is a list of objects row by row in which every

column is a property.

To edit properties the user just needs to edit the cells without having to click in an icon every time he wants to change a property, than after editing everything desired, the user just needs to click in the Update Database button that will verify if everything is correct to be updated and then it updates, if that is the case. The user can also create objects just by adding a value to the cell after the last row with a value, in the web app you need to click a button and then insert a name in a popup.

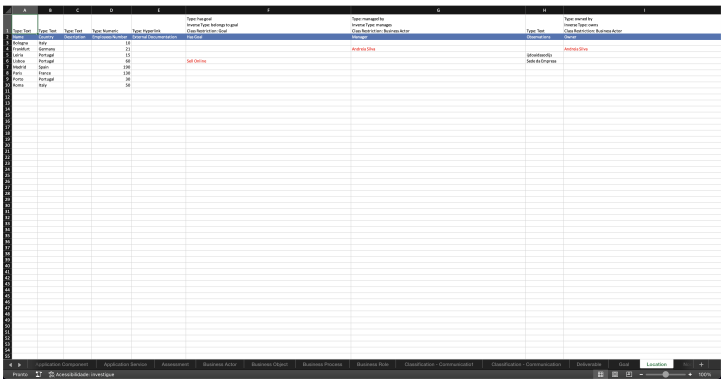

Figure 5.4: Excel ATLAS: A sheet with the information of the Location class

So as we can see in terms of usability, my solution is a lot more practical and takes a lot of less time to edit a number of objects than using the ATLAS web app, the design of the web app is a lot more appealing than my solution, but using a program like Excel does not let me do designs like in a web app.

5.1.2 Efficiency

In this section, we will check which solution is more efficient, I will use a table with different tests and the time both solutions took to see each one is faster.

The table shows only the time it took to edit the objects, because when you run the update database function in my solution, the time it takes to run will be highly dependent in the amount of data you have imported in the Excel sheet, so it is not easy to test how many time it would take.

Table 5.1: Efficiency Tests Table

Test Details	Excel ATLAS Time	ATLAS Web App Time
Alter 6 properties in the same object	36s	40s
Alter 3 property of 8 different objects of the same class	55s	1m56s
Alter 3 property of 3 objects of different classes	1m3s	1m32s
Create 6 new objects of the same class	13s	31s
Create 2 new property, one of type text, and another of type reference	26s	37s

From these results, we can observe that my solution is easier to edit the data, because it is really simple, you do not need to always be opening popups and it is all in one tabular page. The time it then takes to update the database will depend on the amount of data inserted into Excel. But with these time

differences in editing the data, it will easily be faster even with the update function having to do a lot of verification's to check everything is correct and then as it uses Python is always going to be slower than the languages used in website's technology.

It is necessary to remember that these results are always going to depend a lot on the performance of the computer used, the speed of the Internet, and some other external factors, so the same tests I did here could have been completely different if tested in another computer, but even if it was used a slow computer or slow internet, I believe that the difference of the times would be even more beneficial for my solution.

5.2 Pros and cons from Excel

As every other development decision, there are advantages and disadvantages to using Excel for this solution, as an advantage you have a very good program for a tabular view, with already very good features for data editing and analysis, as disadvantages Excel allows the user to change everything, so it is really hard to do an interactive interface, so the user cannot crash by editing the wrong cells. The fact that the users will be all just employees of the Link Consulting company helps, so its known that they are qualified to use the system.

Advantages:

- **Very known and commonly used program**: Microsoft Excel has existed since 1985, most computer users are familiar with Excel.
- **Intuitive Interface**: As Excel is such a known program, it is really easy to use it, even if you as user do not know how to do something, there is always a lot of tutorials online, so everything becomes very intuitive.
- **Easy to do charts from the data**: Excel has features to create charts from the data in the sheets, the user selects a type of chart and then chooses the cells with the data for the chart.
- **Use formulas to analyse data**: The formulas Excel has, helps the user do some mathematical calculations, those calculations can be useful to make statistics of the data, getting average numbers, max, min and many other formulas.
- **Other features that make it easy to edit data**: Excel has also very interesting features to edit data, it has a suggestion system that can help rewrite values that are similar or alike, there is also the drag option to copy, where it can detect patterns and will continue the pattern for the cells you dragged.

Disadvantages:

- **Gives a lot of freedom to the user**: Normally, giving freedom to the user is a good thing, but in our case, since I built a system in which some cells cannot be edited without the system having some bugs, it is a problem for me that the user can edit everything.
- **Users need to know how to use the program, need to read the user manual**: A first time user maybe would have some trouble understating how the inputs and buttons work, that is why I created a user manual.
- **The user needs to have Excel and Python installed**: With a web application, there is no need to have anything installed on the computer, but with this solution the user must have Excel and Python installed to use it.
- **Does not have a very intuitive design**: It is difficult to create a intuitive and attractable design in an Excel sheet, so the design is very simple but very practical also in my opinion.

5.3 Efficiency Tests

5.3.1 Setup Enviroment

Here I will show the setup I used to do these tests:

- MacBook Pro (16 inches, 2019)
- CPU: 2,6 GHz Intel Core i7 of 6 cores
- RAM: 16 GB 2667 MHz DDR4
- GPU: AMD Radeon Pro 5300M 4 GB

5.3.2 Import data from Database

To test the importing from the database, I created 10 classes, each class with 50 similar objects with only Name property, and I will import different amounts of classes and see how much time the system takes.

This values are not definitive, this means that the values vary from test to test, but the values are always around this numbers, we can see the system takes at least 4 seconds to be able to import one class and then importing each one has a difference of around 0,30 seconds, of course this happens because the classes are all similar with the same very simple objects, otherwise each class would have a different import time.

Table 5.2: Import Performance Tests Table

Test Detail	Time
Import 1 Class	4,44s
Import 2 Classes	4,71s
Import 3 Classes	4,79s
Import 4 Classes	5,15s
Import 5 Classes	5,38s
Import 10 Classes	7,67s

5.3.3 Update data to Database

Here I will do 3 different tests: Creating objects, Creating Properties, and Editing Properties.

Create Objects

To test the creation of objects I will do three different environments, one where Excel only has the class we are creating the objects, one where it has extra 100 objects with two extra classes of 50 objects that are not being changed, and one with 300 objects with six classes of 50 objects that are also unchanged.

Table 5.3: Create Objects Performance Tests Table

Test Detail	Time with no extra objects	Time with extra 100 objects	Time with extra 300 objects
Create 1 Object	6,4s	7,39s	8,02s
Create 2 Objects	7,25s	8,77s	9,11s
Create 3 Objects	8,51s	9,47s	9,65s
Create 5 Objects	9,43s	10,1s	10,99s
Create 10 Objects	13,45s	13,74s	14,32s
Create 30 Objects	27,33s	28,12s	28,68s

We can see that running an update is always going to take at least around 6 seconds, taking into account that after every update, all the classes that were previously imported will be imported again to ensure everything is well updated, so that takes around 4 seconds according to the results of the table Import Performance Tests Table, so 2 seconds to do the verifications and edit the database. We can see that the extra objects to verify take sometime to be verified, even though nothing is altered in them.

Create Properties

In these tests I will only test with just the class we are editing, there is no need to test with extra classes again, because the results would be the same as in the previous tests. There will be two different tests, one where all the properties will be of type Text and another of new references, so the system will have to create the reference type.

We can observe again that it takes a minimum of 7 seconds to run an update to create properties

Table 5.4: Create Properties Performance Tests Table

Test Detail	Time of type Text	Time of type Reference
Create 1 Property	7,03s	7,61s
Create 2 Properties	7,22s	9,48s
Create 3 Properties	7,52s	10,07s
Create 5 Properties	8,67s	13,1s
Create 10 Properties	11,23s	19,66s

and every extra property will only take around more 0,20 to 0,30 seconds, this for a regular type, when creating references types it takes longer, because the system needs to verify everything is correct, create the new basetype, and then create the property.

Alter Properties

To alter the properties, I am going to use the classes I created to test the imports, so the classes with 50 objects. I will alter one of their property, each test with a different number of alterations, and each column is the amount of classes in which I alter.

Table 5.5: Alter Properties Performance Tests Table

Test Detail	Time with 1 Class	Time with 2 Classes	Time with 5 Classes
Alter 1 Property	6,77s	7,11s	8,93s
Alter 2 Properties	7,06s	8,09s	11,05s
Alter 3 Properties	7,18s	8,54s	13,31s
Alter 5 Properties	8,25s	9,87s	17,32s
Alter 10 Properties	9,62s	14,14s	27,05s
Alter 30 Properties	18,95s	29,82s	1m 9,25s

As in the other tests, an updates takes around 6 to 7 seconds, and every other change takes 0.3 seconds, we can see that the times are fairly linear to the amount of objects we are changing.

In my opinion the performance is not very good, because taking 27 seconds to create 30 objects or even 1 minute and 9 seconds to edit 150 properties is too much time, this is one of the areas that should be improved in the future.

6

Conclusion

Contents

6.1 Conclusions

I started by trying to find an open source program that would make charts from a database and as the database is not a common relational database, it would have been harder to find a program for this purpose. After an unsuccessful search I realized it was needed to do a program that would do the charts from the ATLAS API. Excel came up as a very good option because it would let us import the database into a very commonly used tabular view and it would be really easy to make charts from the data, due to the already implemented features of Excel. Then we realized that it was a good opportunity to also create a program that can be used to edit the data in the database.

I started by developing the Python scripts that interact with the ATLAS API and get the data from the database, format the data to print in Excel, store the necessary information in a JSON file to then be able to compare the difference if the user wanted to update the database, after that I did the update script which compares the original Excel imported with the one that was edited by the user, over the time this script evolved by making more verifications for errors or adding more features, like creating new objects, or creating new properties.

After having a first version of the Python scripts, I started developing the VBA code to call the Python scripts and getting all the Excel functions to work. When a first version of the program was done, we had a feedback session where it was suggested to have a way of only importing some classes and some properties, instead of all at the same time, and with that I developed the Python script that gets the names of the classes and properties so that the user can choose what to import in Excel.

Then I developed the system to create charts from the data imported, the code for this functionality is all done in the VBA, it will get the data from the sheets and will create a chart using the info selected by the user.

After all the functionalities were done, I developed the error display system to help the user know where they are making mistakes, I did some polishing of the code, did some testing to make sure everything was working properly, then I did a Windows and a Mac version because some functionalities of the VBA do not work in the same way for the different operating systems, to end, I did an installer for the program and a user manual to help the users learn to use the system.

This thesis was highly educational for me, I was able to learn a new programming language, and I learned to do an installer. It was very important to me that the system I was developing was actually going to be used by users, which normally does not happen on the other projects in the IT courses, it made me have to optimize and polish everything, do extensive testing, and make an error display system.

6.2 System Limitations and Future Work

I believe that no program is ever perfect, there is always something that can be added or something that can be upgraded. In every software project it is very important to set a time limit, otherwise the developer will be working on the project forever, always adding some feature or optimizing some other feature.

In my opinion the biggest limitation of my solution is that it takes some time to run the API calls and the python scripts. Python is already not recognized as a performance language, so it always takes a little longer than most other languages, and then the API calls are not optimized for this program, so if they were, the program would become exponentially much faster.

There is also the question of the concurrency, I mentioned in Concurrency Problems, there can be some problems when someone else edits the repository, in between the user imports to Excel and updates the repository. For this, I think it needs to be analyzed the trade-offs mentioned before in the section and check what is better for the solution security/performance.

Here are some new features and upgrades that the system could receive in the future in my opinion:

- **Making a design for the Control Panel sheet**: At the moment, the design of the Control Panel is very simple, but Excel is capable of doing some interesting designs with buttons, images, and more, it would be nice to have a more interactive design.
- **Restricting the cells the user can edit, for safety of the program**: To ensure that the program runs without errors, it would be good to lock all the cells that are not supposed to be edited in the Control Panel, because otherwise the user can create some bugs, this problem is mitigated by the fact that the users are all Link Consulting employees, so they are qualified to use the program.
- **Doing more efficient API calls specific for this program**: As I mentioned earlier, the API calls are not optimized for this program, which means that API sometimes gets too much unnecessary information, there are some API calls that need to be done class by class, and it would be much faster if there were a call for all classes. This change would be critical to improve the performance of the code, because API calls are the part of the code that takes the most time.
- **Deleting objects**: For better usability of the program, it would be nice to have an option to delete objects, it could be done by erasing the whole row of that object or with a special string to indicate to erase the object.
- **Deleting properties**: Deleting properties would also be very good, so the user does not need to go to the web app to do so, it could be done by erasing the column with that property or having a special string again to erase the column.
- **Create classes**: Another feature that could be added in the future would be to create classes by

creating new sheets in the Excel workbook, the name of the class would be the name of the sheet, to create a class the only required parameter is the name.

- **Delete classes**: To delete classes, the user would erase the sheet of that class, but it would be better to ask for confirmation, because the user could do it by accident and losing an entire class by accident would not be good.

Bibliography

[1] L. Consulting. Atlas - link consulting. [Online]. Available: https://linkconsulting.com/what-we-do/products/atlas/#CTA

[2] Guru99. What is a database? definition, meaning, types with example. [Online]. Available: https://www.guru99.com/introduction-to-database-sql.html

[3] Hevo. Working with hierarchical database systems simplified 101. [Online]. Available: https://hevodata.com/learn/hierarchical-database-systems/

[4] C#Corner. What is a network database. [Online]. Available: https://www.c-sharpcorner.com/article/what-is-a-network-database/

[5] C. B. Research. Network database model - computer business research. [Online]. Available: https://www.computerbusinessresearch.com/Home/database/network-database-model/

[6] MongoDB. What is an object-oriented database? — mongodb. [Online]. Available: https://www.mongodb.com/databases/what-is-an-object-oriented-database

[7] T. Target. What is a relational database? [Online]. Available: https://www.techtarget.com/searchdatamanagement/definition/relational-database

[8] Guru99. Nosql tutorial: What is, types of nosql databases example. [Online]. Available: https://www.guru99.com/nosql-tutorial.html

[9] A. Developer. Introduction to applescript language guide. [Online]. Available: https://developer.apple.com/library/archive/documentation/AppleScript/Conceptual/AppleScriptLangGuide/introduction/ASLR_intro.html

[10] DbVisualizer. Sql client and editor - dbvisualizer. [Online]. Available: https://www.dbvis.com/

[11] SQLGate. Sqlgate - the most intelligent ide for database. [Online]. Available: https://www.sqlgate.com/

[12] Postico. Postico 2. [Online]. Available: https://eggerapps.at/postico/

[13] Superset. Creating your first dashboard — superset. [Online]. Available: https://superset.apache.org/docs/creating-charts-dashboards/creating-your-first-dashboard/#creating-charts-in-explore-view

Appendix

A.1 Error Display Table

Table A.1: Error List

Error Name	Error Description
Error: No repository name was given	Happens when the user tries to import a repository without giving an actual repository name to import
Error: No username was given	Happens when the user tries to use one of the functions that call the API without giving an actual login username
Error: No password was given	Happens when the user tries to use one of the functions that call the API without giving an actual login password
Error: No mode was given	Happens when the user tries to call an update but does not choose an update mode to use in the API
Error: The username or password are wrong	Happens when the user tries to use one of the functions that call the API without using a correct username and password
Error: There is no repository with that name	Happens when the user tries to import a repository but the repository he inserted does not exist
Error: There is no repository or profile with that name	Happens when the user tries to import a repository but the repository or the profile he inserted does not exist
Error: No Classes with objects to import	Happens when the user tries to import certain classes from a repository but does not choose any
Error: No properties selected on the ... class	Happens when the user tries to import a class with certain properties but does not choose any
Error: Import List does not have a correct option	Happens when the user tries to import certain classes and properties but does not use a correct Option
Error: No update to be done	Happens when the user tries to update the database, but did not do any alteration in the data
Error: (Sheet/Cell) It is expected a reference but is not a reference	Happens when the user inserts anything that is not a reference in a cell that is expecting a reference
Error: (Sheet/Cell) The reference class with the name ... does not exist in this repository	Happens when the user tries to reference one class that does not exist
Error: (Sheet/Cell) The reference class with the name ... is not in the class restrictions	Happens when the user tries to reference one class that is not in the class restriction of that property
Error: (Sheet/Cell) Invalid Numeric value	Happens when the user tries to write something that is not a number in a numeric cell
Error: (Sheet/Cell) Invalid Boolean value	Happens when the user tries to write something that is not a boolean in a boolean cell
Error: (Sheet/Cell) Invalid Date	Happens when the user tries to write something that is not a date in a date cell
Error: (Sheet/Cell) Invalid Hyperlink	Happens when the user tries to write something that is not a hyperlink in a hyperlink cell
Error: (Sheet/Cell) An object on the ... sheet does not have a name property	Happens when the user erases a name property from an object
Error: (Sheet/Cell) It is not possible to alter the basetype information of an already created property	Happens when the user tries to edit the basetype information of an already created property which is not permitted
Error: (Sheet/Cell) It is not possible to alter the name of an already created property	Happens when the user tries to edit the name of an already created property which is not permited
Error: (Sheet/Cell) The basetype information is not correctly formulated, insert a Type:	Happens when the user tries to create a new property but does not use correct basetype information
Error: (Sheet/Cell) There is no name for the new property	Happens when the user tries to create a new property but does not insert a name for it
Error: (Sheet/Cell) One property you are trying to create, has a non standart basetype but does not have an inverse	Happens when the user tries to create a new property that is a reference but does not insert a inverse type for the reference
Error: (Sheet/Cell) The inverse type with the name ..., already exists, change it's name to a basetype that does not exist	Happens when the user tries to create a new property but uses a inverse type that already exists which is not permitted
Error: (Sheet/Cell) The class restriction with the name ... does not exist in this repository	Happens when the user tries to create a new property that is a reference and uses a class restriction with a class that does not exist
Error: (Sheet/Cell) The new object does not have a name property	Happens when the user tries to create a new object and does not insert a value to the name property