2019-12-13

# Database Access Point Security (DAPS)

Janette L. Fong
*Worcester Polytechnic Institute*

Matthew B. Kornitsky
*Worcester Polytechnic Institute*

Yuanda Song
*Worcester Polytechnic Institute*


WPI

BNP PARIBAS

# Leveraging a Three-Tier Architecture to Restrict Direct Database Access from BNP Paribas' Internal Applications

A Major Qualifying Project report to be submitted to the faculty of
Worcester Polytechnic Institute in partial fulfillment of the requirements
for the Degree of Bachelor of Science

**Project Team:**

Janette Fong

Matthew Kornitsky

Yuanda Song

**Project Advisor**

Professor Wilson Wong

Department of Computer Science

**Project Co-Advisor**

Professor Michael Ginzberg

Department of Business

# Abstract

BNP Paribas has adopted an IT security policy that prohibits direct access to company databases from client applications. To ensure that the company remains compliant with security standards, our team implemented a low-maintenance, highly-scalable, authentication service that serves as the mediator between client applications and databases. This middleware interacts with databases to process and relay data, as well as ensures that users are authenticated and authorized to perform the actions that they request. Our team simultaneously developed a responsive, user-friendly web application that system administrators can use to manage user roles for the authorization process. These services are anticipated to serve as the basis for the solution that BNP Paribas deploys company-wide.

# Acknowledgments

Our team would like to acknowledge the following organizations and individuals that provided us with critical assistance and guidance throughout the entirety of our project.

We would first like to thank our advisors, Professor Wilson Wong, Professor Michael Ginzberg, and Professor Robert Sarnie. Having several advisors, each with their own specialties and domain knowledge was truly invaluable. They graciously contributed their time and insight to help us carry out a successful project in the corporate world. Our team would not have been able to achieve our goals without the assistance of these professors.

We would also like to acknowledge the tireless effort put forth by our supervisors at BNP Paribas, Kunal Changela and Pam Nithikasem. Their knowledge of company ecosystems and bureaucratic processes ensured that our team performed in the most efficient ways possible. They also always managed to find time in their busy schedules to help us resolve technical issues and to critique our solution. Kunal and Pam allocated countless hours above and beyond their normal working days to provide our team with access to the resources necessary to complete this project.

Finally, we would like to express gratitude to Andrew Clark, the Head of Global Markets Front Office IT, for arranging this enriching opportunity and to BNP Paribas for hosting us for the duration of the project.

Without the support of these individuals, our project would have been impossible to complete. Thank you all for contributing to the current and future success of this impactful project!

# Table of Contents

# Table of Figures

# Table of Tables

# Chapter 1: Introduction

BNP Paribas is a leading bank in the eurozone and an acclaimed banking institution ("About the Group", n.d.). As of December 2018, the company is located in 72 countries with 202,624 employees across all locations ("About the Group", n.d.). Our team worked at BNP Paribas' New York office for our project.

BNP Paribas has an IT security policy that prohibits user facing applications from directly interacting with company databases. This means that all applications must have a multi-tier authentication architecture in which the client has to communicate with server-side services to access databases (instead of directly interacting with them).

The company currently has several in-house and vendor applications in New York that are accessing databases directly and therefore breaching the IT application security policy. The applications are written in a range of programming languages, which include C#, Visual Basic for Applications, and Python. The goal of this project was to develop an API that interfaces between the infringing applications and the databases that need to be accessed. BNP Paribas employees from across the New York branch should be able to use this API service, so it needs to be able to handle requests from about thirty to one hundred concurrent users.

# Chapter 2: Background and Literature Review

## 2.1 BNP Paribas

Following the establishment of the first entirely mobile bank, Hello bank, in 2013, BNP Paribas (2016a) reflected on its nearly two centuries of growth through an industrial period. The company reminisces on its development from several French banks in the 19th century. The timeline begins with the establishment of one of BNP Paribas' forerunners, Société Générale de Belgique, in 1822. Two other ancestors of BNP Paribas -- Comptoir National d'Escompte de Paris (CNEP) and Comptoir National d'Escompte de Mulhouse – were established after a French economic downfall and political revolution in 1848. In 1932, Comptoir National d'Escompte de Mulhouse collapsed and re-emerged as Banque Nationale pour le Commerce et l'Industrie (BNCL) (BNP Paribas, 2016a). These banks continued to expand across the world throughout the mid-20th century, going through collapses and reestablishment. In 1966, CNEP and BNCL merged to become BNP, for 'Banque nationale de Paris' (BNP Paribas, 2016a). The 21st century brought the digital revolution, which caused challenges for the banks to catch up with the changing world. As a result, in 2000, BNP and Paribas merged to form the BNP Paribas Group (BNP Paribas, 2016a).

In the United States, BNP Paribas organizes its businesses into two main categories: Retail Banking & Services (RBS) and Corporate Institutional Banking (CIB) ("Activities", n.d.). RBS includes both branch networks and a range of other specialized financial services offered to the general public ("BNP Paribas in the U.S.", n.d.). CIB provides capital markets, securities services, financing, treasury, and advisory solutions ("BNP Paribas in the U.S.", n.d.).

BNP Paribas (2016b) highlights its goals of setting European banking standards with global reach and increasing access via the Internet. In pursuit of this global reach, the company has established a customer base of 32 million individuals and 850,000 entrepreneurs, professionals, enterprises, and corporate clients. As a result of its continual growth, BNP Paribas managed to reach 2.04 trillion euros in total assets, 42.516 billion euros in revenue, and 8.005 billion euros in net income ("Consolidated Financial Statements", 2019).

To support the development of new technologies that reinvent banking and financial services, BNP Paribas (2016c) established a model for collaborating with fintech start-ups. The model is centered around four pillars: access to banking services, co-creation, acceleration, and investment. In support of the first pillar, access to banking services, BNP Paribas strives to

support start-ups with their development around the world. In accordance with the co-creation pillar, BNP Paribas works directly with its customers to develop solutions that benefit both parties. The third pillar facilitates this co-creation through the international BNP Paribas Plug and Play program, which involves a partnership between some of the world's largest start-up incubators. Lastly, the investment pillar encompasses direct investment in start-ups by virtue of complete acquisition, equity interests, or third-party funds.

## 2.2 Multi-tier Architecture

Software architecture refers to the fundamental organization of a software system. It details the various components of a system and presents how those constituents interact with each other to accomplish a specific function or set of functions (Eeles, 2006). A multi-tiered or multilayered architecture follows the client-server model in which the functionality associated with presentation, application processing, and data management are isolated. In the client-server model, a distributed application partitions workload between two communicating nodes, namely the requestor (client) and the provider (server). Client programs request data from servers and process the responses in order to provide a service to the user. The server synchronizes and manages access to the resources that the clients require in order to perform those services ("The Client/Server Model", n.d.). The N-tier architecture is a flexible and reusable model that enables developers to add additional boundaries to their application for purposes of scalability, reliability, and security.

### 2.2.1 Two-Tier Architecture

As seen in Figure 2.1, a two-tier architecture is a straightforward software architecture that consists of a presentation layer that sources information from a server or data layer ("What is Two-Tier Architecture?", n.d.). The presentation layer, also known as the client layer, is the topmost layer of an application. It is the layer that the user sees and interacts with to communicate with other layers of the application. Likewise, the data layer interfaces directly with an application's database and uses commands to extract data that the client requests. Due to its simplicity, the two-tier architecture facilitates the construction of robust applications for homogeneous environments. The close proximity of the client and data sources offers users higher performance compared to architectures with additional layers. On the other hand, this

model requires the majority of the application logic to reside on the client-side. This places limitations on a vendor's ability to push crucial updates to clients and overall makes controlling software versions a difficult and potentially expensive process ("What is Two-Tier Architecture?", n.d.).

The two-tier model also lacks scalability. Since there is no middleware to reroute requests to various database instances, server resources must be divided among the various concurrent users. Similarly, without a central authentication service, users may require separate credentials for every server that they need to access. Overall, the two-tier architecture is advantageous for small and rarely changing applications but becomes difficult to manage as the user base and number of updates increase.

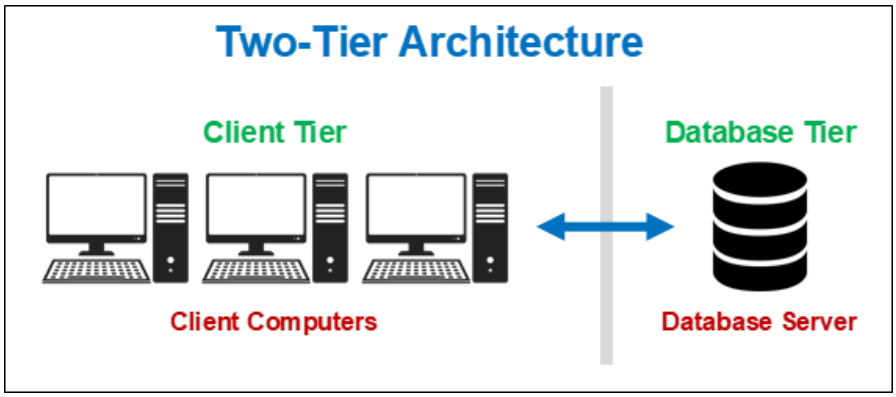


*Figure 2.2.1 - Two-Tier Software Architecture (Rajkumar, 2017)*

### 2.2.2 Three-Tier Architecture

A three-tier architecture is the most widespread of all the multi-tier architectures. Depicted in Figure 2.2, in addition to the presentation and data layers, a three-tier architecture incorporates an application layer (Rajkumar, 2017). The presentation and data layers serve much the same purpose that they do in a two-tier architecture, but the application or business logic layer serves as a mediator between the two outer layers and handles any intensive data processing that an application may need to perform.

Although the additional node may increase maintenance expenses and users may see a reduction in performance, a three-tier architecture has many advantages over a two-tier architecture. First, the overall security of an application will be enhanced. The application layer

helps to curtail data corruption and unauthorized database procedures by ensuring data validity and preventing clients from interacting directly with an application's databases ("What is Three-Tier Architecture", n.d.). A three-tier architecture is also more scalable because the application layer can efficiently distribute traffic among all the servers in a database cluster. Moreover, the business logic and database structure can be modified without changes to the client. Since the middle tier receives requests and returns data, vendors can update all the processes in between those operations without being concerned about whether users will install updates or patches (Rajkumar, 2017). In summary, a three-tier architecture offers security and scalability advantages over a two-tier architecture but is generally more costly and can sacrifice performance.

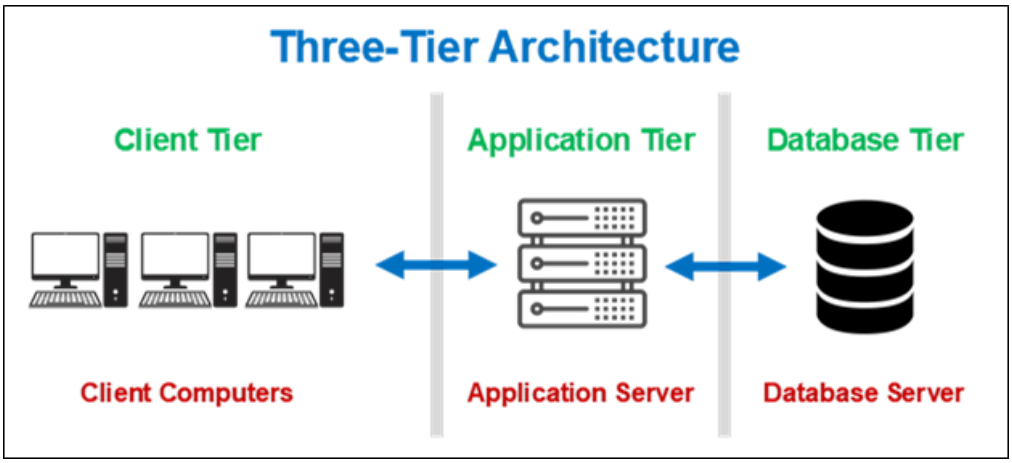


*Figure 2.2.2 - Three-Tier Software Architecture (Rajkumar, 2017)*

## 2.3 Hosting vs Self-Hosting

Applications can either be hosted or self-hosted. Both options have advantages and disadvantages, and it is important to understand their differences to ensure that confidential information is not leaked into the public domain.

### 2.3.1 Hosting

Hosting refers to an application that utilizes an external host to deploy an application to the Internet. These external hosts provide technologies and services that are necessary to make the application globally accessible. Trevellyan (2017) notes that using a hosting platform means that the host will handle any software updates, which may reduce the burden of maintenance.

Trevellyan's analysis of hosting services also reveals that despite reduced maintenance, using a hosting platform has inherent restrictions on functionality and customization. Since the application is dependent on the hosting platform, if the hosting company goes out of business, the application will need to be hosted by another source (Trevellyan, 2017). Some examples of hosted applications include Google Docs or Gmail, which are both hosted on Google's servers (Olic, 2017). Using an external hosting platform involves accessing the Internet to share information from the application, so hosting an application with sensitive or confidential data may not be the best option.

### 2.3.2 Self-hosting

In contrast to hosting, self-hosting is when the application's creator installs and accesses software from the creator's own server (Olic, 2017). ActiveCollab (2019), the creators of a self-hosted project management software, promotes the flexibility of self-hosted services that arises from complete control over applications and their data. ActiveCollab acknowledges the need for developers to handle software updates and the maintenance of their self-hosted platforms. While hosting platforms are not customizable, self-hosting platforms are because the creators are in control of the entire system. Another advantage of a self-hosted platform is that they are completely private (Olic, 2017). If the information from the application is sensitive and is not meant to be shared across the Internet, then using a self-hosting platform is the appropriate solution.

# Chapter 3: Methodology

The goal of this project was to build a multi-tier architecture where clients communicate through a mediator to access databases, instead of accessing them directly. As a supplementary component of this middleware, a graphical user interface that allows administrators to manage the permission roles of the users was also constructed.

To achieve this goal, our team had the following objectives:

1. Review BNP Paribas' current data access model
2. Industrialize the current solution, test query return times under heavy loads, and deploy the infrastructure to a remote server
3. Manage the privileges of connected users via a graphical user interface
4. Introduce developer teams to the new connectivity protocols and showcase how client applications written in different programming languages can utilize the service
5. Train Applications Support Analyst to support and maintain the permissions model

In order to achieve these objectives, our team investigated two mainstream software engineering methodologies, the Traditional Software Development Life Cycle (Waterfall Model) and Agile Software Development (Scrum and Kanban).

## 3.1 Traditional Software Development Life Cycle (Waterfall Model)

The Waterfall Model was the first model that was widely used in the software industry (Sarycheva, 2019). It represents a unidirectional flow of software development. In the Waterfall Model, development moves to the next stage only when the previous stage is fully completed. Therefore, the output of one stage acts as an input for the next stage. This model got its name because of its sequential and steadily downwards flowing nature.

Figure 3.1 shows the unidirectional characteristic of the Waterfall Model. The sequential phases in the Waterfall Model are as follows:

1. **Requirement Gathering and Analysis**: The team captures all the requirements of the system and delivers a requirement understanding document.
2. **System Design**: The requirements for understanding documents are studied. In this phase, the team is designing how the requirements will be technically implemented. This phase covers programming languages, data structures, and different tools being used in the project.
3. **Implementation**: Translating the models into code - all models, business logic and service integrations that were specified in the System Design phase should be implemented. Unit testing also occurs in this phase.
4. **Integration and Testing**: All implemented requirements are tested and integrated into a system. Post integration testing is also performed to find defects in the system.
5. **Deployment**: Once the functional and nonfunctional testing is finished, the product is deployed to the respective environment.
6. **Maintenance**: The final phase contains fixing the issues after deploying to the respective environment and enhancing the product.

The traditional software development life cycle is simple to understand and follow, making it suitable for smaller projects. Nevertheless, because no working software is produced until the latter stages of the development life cycle and it is difficult to measure progress within the stages, the waterfall method is typically suboptimal for substantial software projects.

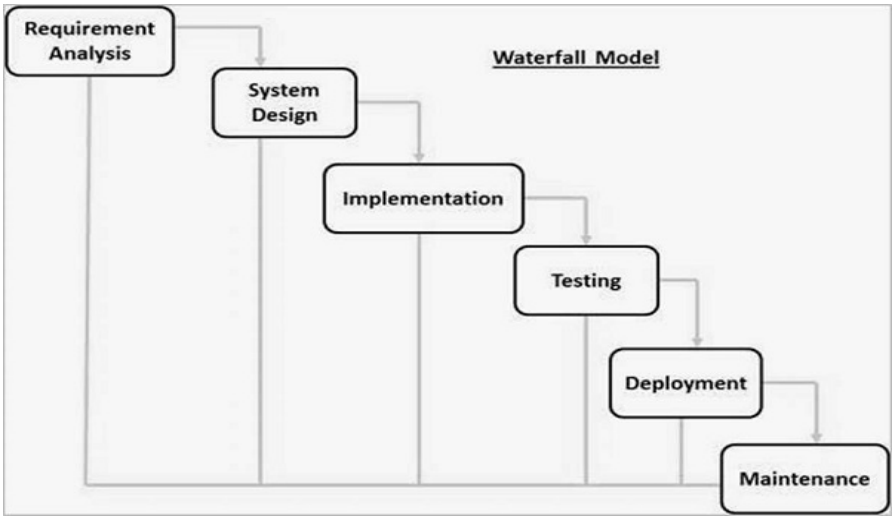


*Figure 3.3.1 - The Waterfall Model*

## 3.2 Agile Software Development

Agile software development is a software development approach that emphasizes incremental delivery, team collaboration, and continual learning, instead of delivering the final product at the very end of the development cycle (Visual Paradigm, 2019)

Agile, however, is not a specific methodology. It is the way the team thinks and acts. Generally, an agile team breaks a larger project into smaller pieces called user stories, which describe product functionality, help prioritize smaller tasks, and ensure that functional components are delivered at the end of each iteration (usually two week-cycles). Differing from the unidirectional Waterfall Model, Agile Software Development is an accumulated and iterative approach.

There are many frameworks under the Agile Software Development category, as shown in Figure 3.2. Our team looked at the two most popular: Scrum and Kanban.

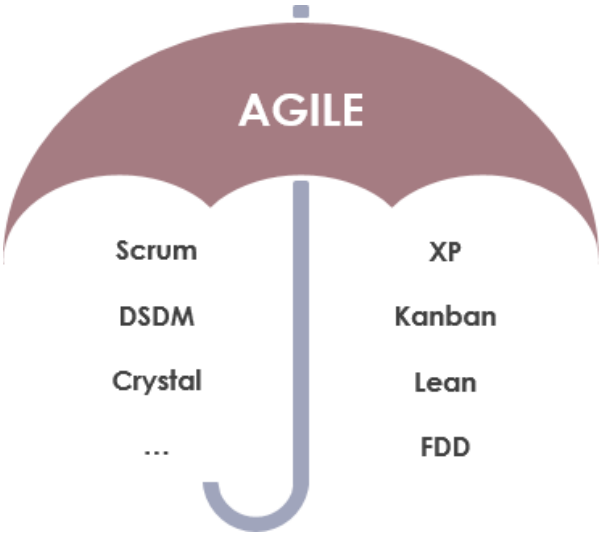


*Figure 3.2 - Agile Development Frameworks*

### 3.2.1 Scrum

Scrum is the most popular framework under the Agile Software Development category. A Scrum team relies on self-organization, in which the team does not have a specific leader to guide the team toward its goals. A Scrum team focuses on an adaptive product development strategy, where the cross-functional team members work as a unit to achieve a common goal in a short “sprint” (Visual Paradigm, 2019). Within a Scrum team, there is a product owner and a

scrum master. The product owner is responsible for the product, managing the project backlog, and deciding the progress of the project. The scrum master is responsible for ensuring the team understands the scrum process and acts as a facilitator of the team.

Meeting on a regular basis is one of the best ways for the team to keep track of the project and get synchronized between the team members. Fortunately, Scrum incorporates frequent brief meetings in the form of four ceremonies: sprint planning, daily scrums, sprint reviews, and sprint retrospectives. The sprint planning meeting takes place at the beginning of each sprint iteration and allows the team to determine the user stories to be completed by the next sprint iteration. The short daily scrum meetings held every morning allow team members to synchronize the previous day's work and set a plan for the current day. At the end of each sprint, the retrospective and review meetings allow the team to look back at the completed stories in the sprint and reflect on what can be improved for the next iteration.

To keep track of a team's progress, a burndown chart is used to compare the number of user story points needed to complete the project compared to the number of sprints left. The visualization is also used to help the team keep track of the velocity of completing user stories in each sprint iteration. This serves as an indicator of whether the team is on the right track and if completing all the user stories is feasible within the given timeframe.

### 3.2.2 Kanban

Kanban is another Agile Software Development framework that helps the team visualize their tasks, maximize efficiency, and be agile (Kanbanize, 2019). The Kanban board has different columns to represent the workflow of a project. For example, some Kanban boards classify tasks as one of "To Do", "In Progress" or "Completed". For the Agile team, cards that lie under the columns are the user stories that the team is going to implement. The team will have a backlog for project ideas that can be picked up when they are ready. The final element of Kanban is the delivery point, where the team takes delivery of the product or service to the customer.

## 3.3 Methodology Selection and Timeline

Our team adopted Scrum's agile workflow and Kanban's visualization to help maximize productivity as well as track the progress in this 7-week project. We broke down the epics,

mentioned in Section 5.2, into smaller user stories and eventually specific tasks. We used the Fibonacci scale to weigh each task according to the estimated time it would take to complete. Based upon the rating of each task, the team decided to divide the term into weekly sprints and created the project timeline depicted in Figure 3.3.

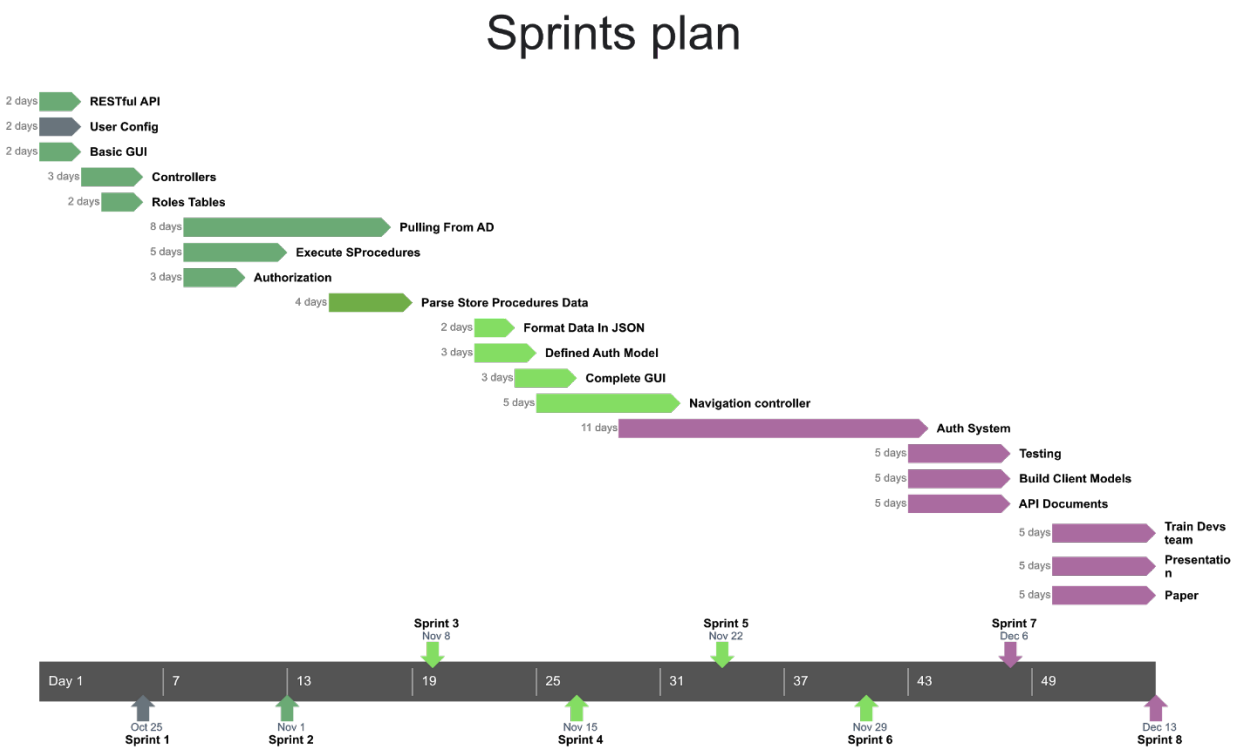


*Figure 3.3 - Sprint Plan*

### 3.3.1 Manifesto for Agile Software Development

The Agile Manifesto contains four guidelines: Individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan (Agilemanifesto, 2019). Our methodology selection followed these values.

1. **Individuals and Interactions Over Processes and Tools**: The team valued individuals and interactions more than the process and tools. The scrum methodology is a popular framework that gave us a high-level understanding of how to develop our solution.

Overall, the key to the scrum is to bring individuals together to communicate and work on a common goal.

2. **Working Software Over Comprehensive Documentation**: Our team used Scrum in combination with Kanban to ensure that we had a functioning product that at the end of each iteration. Comprehensive documentation was not suitable for our development since it is time-consuming. Documentation for this project was not completely disregarded, however. In addition to documenting common problems and solutions, we also provided informative READMEs describe how developers can use and modify our deliverables.

3. **Customer collaboration over contract negotiation**: Customer collaboration is one of the essential aspects of the Scrum methodology. Our team had weekly meetings with our sponsor to reported progress and incurred obstacles. These meetings enabled the team to receive feedback on our solution and provided guidance to direct subsequent steps of development.

4. **Responding to change over following a plan**: Unlike the traditional software development methodology, where changes are very expensive, the scrum methodology allowed the team to change the priorities of different user stories between iterations. As new requirements presented themselves, our team was easily able to incorporate them in the product backlog.

In order to implement the Scrum workflow and with Kanban visualizations, our team had a weekly sprint planning meeting at the beginning of each Monday, daily scrum meetings, and weekly sprint retrospective reviews at the end of the week. In the sprint planning meeting, the team selected a reasonable number of user stories from the backlog to implement within the one-week sprint. The user stories were then formulated into tasks that each team member ultimately chose to work on. In the daily scrum meetings, our scrum master led the team to discuss what has been done in the previous day and development plan for the present day. These daily scrums also served as an opportunity for the team to discuss any obstacles that may have been encountered.

The sprint retrospective reviews that happened at the end of each week gave the team a chance to look back on what was accomplished and how the team could improve in future sprints.

# Chapter 4: Software Development Environment

There are many possible software products that can be used to develop functional solutions. To ensure that our team was working with the most efficient technology stack, we needed to consider a variety of alternative tools for each component of the development process. This chapter outlines the considerations we made when choosing technologies for management, software development, and data storage.

## 4.1 Project Management Software

### 4.1.1 Bitbucket

To manage all the code that we wrote for our project, we used Bitbucket. Since many source code repository hosts, including GitHub, are blocked by BNP's firewall, we did not have much of an alternative. Any code that we developed for BNP Paribas had to remain inside the company's network and could not be accessible to any parties outside of the organization. Bitbucket offered the vital functionality of a source code repository host, such as branch management, which enabled us to develop on feature branches and merge to the master branch only when the applications were in stable states.

### 4.1.2 Google Drive

Our team used Google Drive to organize documents that were related to presentations or project papers. Google Drive's features of Google Docs, Google Slides, and Google Sheets allowed us to work on documentation and presentations simultaneously.

### 4.1.3 Trello

To organize our tasks and track our progression, we used Trello. We made Trello cards for all the tasks that led to the finished product. Using a Trello board allowed us to discuss which tasks and user stories we wanted to complete each week, and to specify certain tags to prioritize each Trello card. An alternative to Trello that we could have used is Google Sheets, but we decided to use Trello because of the draggable interface that enabled the classification of tasks (e.g. To Do, In Progress, Completed).

### 4.1.4 Slack

Our team mainly communicated through Slack. We chose to use Slack as our communication tool because we could neatly organize different topics of discussion in separate channels and we could highlight important messages. This ensured that specific topics were grouped together, and important messages could be referred to easily. Alternative communication services include Group Me, Discord, Facebook Messenger, and many others, but we decided to use Slack because of its familiarity to our team members.

## 4.2 Integrated Development Environment

For our project, we developed code in the Visual Studio IDE. JetBrains also offers a .NET IDE, however Visual Studio supports a large variety of other application types. BNP Paribas also recommended that we program in Visual Studio so that if any problems were incurred, we could get help from other developers. We installed Visual Studio on our corporate computers, but also had copies on our personal laptops, so that we could continue to develop sample applications outside of work hours.

## 4.3 Software Tools / Programming Languages

### 4.3.1 C#

Our team used C# to develop the middleware and the permissions manager. We decided to use C# because it is what BNP developers are most familiar with, and the company already has a large selection of the client applications that are written in C#.

### 4.3.2 JavaScript Object Notation (JSON)

For the middleware service, we planned to utilize JavaScript Object Notation (JSON). Instead of trying to convert different client application requests, written in a multitude of programming languages, into one universal language that the middleware service could read, we simply used JSON to encode information in HTTP request bodies and responses. We also developed sample applications in C#, Python, and VBA to demonstrate how various client applications would transmit and parse JSON data.

### 4.3.3 .NET Framework and ASP.NET

Since the middleware is effectively an API and the permissions manager is a web application, we decided to develop them using ASP.NET instead of Python, PHP, Node.js, Ruby, Perl, Haskell, or Flask. This decision was primarily motivated by the compatibility between C# and Microsoft services. Since the middleware leveraged Windows Authentication and the permission manager relied heavily on information from Active Directory groups, the choice between ASP.NET and other web frameworks was made quite easily.

### 4.3.4 Internet Information Services (IIS)

To host the middleware and permissions manager, BNP Paribas provided us with Windows 10 servers running Internet Information Services (IIS). Since the majority of the company's applications are hosted with IIS, it was easy to find developers to help us resolve any issues that we had. For local development, we relied on IIS Express because it was built into Visual Studio and provided a straightforward means to test our applications. Once the applications were finalized, however, they were ultimately deployed to the Windows 10 servers running the unabridged version of IIS.

## 4.4 Databases

We would not execute custom queries on BNP Paribas' databases, but there were stored procedures on Microsoft SQL Server and Oracle databases that we executed to obtain the information that client applications requested. Since BNP Paribas already had instances of Microsoft SQL Server and Oracle databases, we did not need to consider alternatives for its main data sources. Additionally, we were provided with access to BNP Paribas' internal database management system known as the Configuration Manager. The Configuration Manager stores information for many of BNP's applications and was regarded as the appropriate database choice for storing role data. Since our team knew nothing about the Configuration Manager, we took the advice that was given to us and incorporated it into our solution. Due to the number of problems that resulted from the use of the Configuration Manager, it was not the optimal choice of data storage (see section 9.4 for more information).

# Chapter 5: Software Requirements

## 5.1 Functional and Nonfunctional Requirements

A requirement is a feature that a system must have or a constraint that must be satisfied in order to meet the criteria outlined by a client. Typically, requirements are partitioned into those that are functional and those that are nonfunctional. Functional requirements define a system's specific range of capabilities, which are independent of implementation (Bruegge & Dutoit, 2010). Nonfunctional requirements, on the other hand, define constraints on the system's operation that are not directly related to its functionality (Bruegge & Dutoit, 2010).

### 5.1.1 Functional Requirements

After several discussions with our manager at BNP Paribas, it became evident that our solution consisted of two interconnected, but distinct components. The foundational constituent of our solution was the API that decoupled user facing applications from company databases. The following functional requirements outline the specifications of the middleware that BNP Paribas envisioned:

1. User facing applications must not communicate directly with databases
2. Users must only execute stored procedures that have been explicitly authorized

The supplemental subdivision of our solution was a graphical user interface that would enable administrators to manage the permissions that the middleware utilizes for verifying the authorization of users. The following functional requirements delineate the specifications of the GUI that BNP Paribas envisioned:

1. Administrators must be able to manage the stored procedures that a user can execute
2. A list of users must be generated automatically from an Active Directory Domain Service
3. Administrators should not manually input data, except for server connection details
4. Given a server connection string and a set of credentials, the names of the databases on the server must be imported automatically, along with their stored procedures
5. Administrators must be able to modify the permissions of many users simultaneously

### 5.1.2 Nonfunctional Requirements

In addition to many general categories of requirements, the FURPS+ model defines usability, reliability, performance, and supportability as the primary classifications of nonfunctional requirements (Bruegge & Dutoit, 2010). With the potential to define BNP Paribas' data access model, our consolidated solution had to be usable, reliable, performant, and supportable.

**Usability** refers to the "ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component" (Bruegge & Dutoit, 2010). Following our departure from BNP Paribas, the company's engineers will need to modify user facing applications to communicate with our middleware, instead of databases directly. Given that there were hundreds or thousands of applications that needed to interface with our solution, setting up a new connection between an application and the middleware had to be quick and straightforward. Likewise, the implementation of the permissions manager had to provide an intuitive interface that would enable administrators to expeditiously modify a user's privileges.

**Reliability** is defined as "the ability of a system or component to perform its required functions under stated conditions for a specified period of time" (Bruegge & Dutoit, 2010). Since our solution interacted with many types of clients and affected the efficiency of company employees, our solution had to be robust. The implementation of the middleware and permissions manager had to appropriately handle invalid or unexpected input and avoid crashing at all costs.

**Performance** is associated "with quantifiable attributes of the system, such as response time, throughput, availability, and accuracy" (Bruegge & Dutoit, 2010). Initially, our system had to be capable of supporting 100 concurrent users, but if implemented company-wide, that number could be multiplied by 3 orders of magnitude. Likewise, the permissions manager had to be capable of extracting data from 100s of databases and displaying information pertaining to all of the middleware's users.

**Supportability** is concerned with "the ease of changes to the system after deployment" (Bruegge & Dutoit, 2010). Technology is perpetually evolving, and companies are constantly changing their systems. Our solution had to be modularized so that it is easily maintained and can be made compliant with the company's ever-evolving standards.

## 5.2 Epics and User Stories

As outlined in the previous section, to meet the requirements and expectations established by BNP Paribas, our solution was divided into 2 parts. Those components can be more formally defined by epics, which by virtue of user stories are mapped to the requirements in section 5.1 to illustrate how end-users will interact with each facet of the application.

***Epic 1:*** *Develop an API that interfaces BNP Paribas' databases, executes authorized stored procedures, and returns data to user facing applications*

a. As a manager
    i. I want to prevent my employees from directly accessing company databases so that I can ensure the data has not been manually modified
    ii. I want to restrict my employees' access to databases so that they cannot view or manipulate data without explicit authorization
    iii. I want a solution that is hosted on a remote server so that employees from across the world can utilize its services
b. As an employee
    i. I want to access company databases quickly so that I have the information I need to make informed decisions
c. As an engineer
    i. I want a solution that is compatible with applications written in many programming languages so that I do not have to create a new service for each type of application
    ii. I want a solution that is easy to troubleshoot and modify if needed

***Epic 2:*** *Construct an application with a GUI that enables system administrators to manage the stored procedures that a user can execute*

*d.* As an administrator
    *i.* I want to manage the stored procedures that a user can execute so that employees cannot view or manipulate data without explicit authorization
    *ii.* I want to edit multiple user permissions at once so that I can save time
    *iii.* I want to limit manual input so that I make fewer mistakes
    *iv.* I want user lists to auto-populate so that I do not have to manually enter information
    *v.* I want database lists to auto-populate so that I do not have to manually enter information
    *vi.* I want stored procedure lists to auto-populate so that I do not have to manually enter information
    vii. I want to be able to access the GUI without having to install an application on my computer
    viii. I want a user interface that is easy to navigate so that I can complete my job quickly
e. As an engineer
    i. I want a solution that is easy to troubleshoot and modify if needed

## 5.3 Use Cases

A use case itemizes the events or actions that are expected when a human or external system is interacting with a specific component of a system. By outlining the flow of events through the system, a use case is intended to encapsulate all possible scenarios for a given piece of functionality. The following use cases depict the anticipated interactions between actors and our solution.

**Use Case #1**

1. Name: Execute Stored Procedure via Middleware
2. Participating Actors: Client Application
3. Entry Conditions:
    a. User is logged into company workstation
    b. The client application sends a POST request to the middleware

4. Exit Conditions:
    a. The client receives output from the stored procedure in JSON
5. The flow of Events:
    1. Middleware parses the request to extract a user ID, server IP address, database name, stored procedure name, and any stored procedure parameters
    2. Middleware queries Active Directory domain controller to ensure that user is in the Active Directory group provisioned for the service
    3. Middleware queries Configuration Manager database for authenticated user's roles to check if the given stored procedure is authorized
    4. Middleware establishes connection to the database server using connection string
    5. Stored procedure is executed
    6. Middleware converts stored procedure output into JSON string
    7. Middleware sends response to client with status code 200: OK and JSON string in response body
6. Alternate Flow of Events:

    [Failed to Parse POST Body]

    1. Middleware sends response to client with status code 400: Bad Request

    [User Authentication Fails]

    1. Middleware sends response to client with status code 401: Unauthorized

    [User Not Authorized to Execute Procedure]

    1. Middleware sends response to client with status code 403: Forbidden

    [Server, Database, or Stored Procedure Does Not Exist]

    1. Middleware sends response to client with status code 404: Not Found

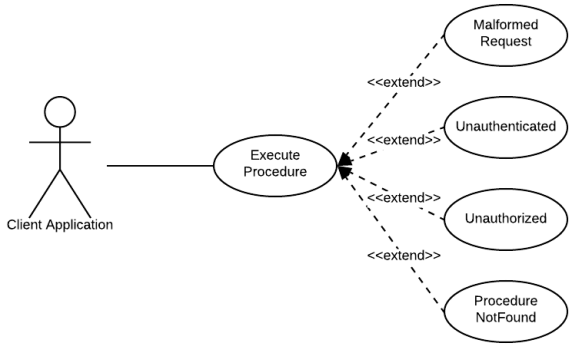


*Figure 5.1 - Use Case: Execute Stored Procedure via Middleware*

**Use Case #2**

1. Name: Add Server to Permissions Manager
2. Participating Actors: Administrator
3. Entry Conditions:
    a. Administrator has been granted permission to use the application
    b. Administrator is in the servers tab of the permissions manager
4. Exit Conditions:
    a. Administrator can view databases residing on the server as well as the stored procedures for those database
5. Flow of Events:
    1. Administrator clicks “Add Connection” button
    2. Form page appears with fields to collect connection details
    3. Administrator enters server type, server IP address, port number, and authentication credentials
    4. Permissions manager establishes connection to server using Windows Authentication or credentials
    5. Permissions manager queries server to get database names
    6. Permissions manager queries databases to get stored procedure names
    7. Permissions manager displays sever, database, and stored procedure names in directory navigator
6. Alternate Flow of Events:
   [Server Does Not Exist]
    1. Display error message to notify administrator that the server does not exist
   [Could Not Authenticate with Server]
    1. Display error message to notify the administrator that the permissions manager could not establish a connection to the database server

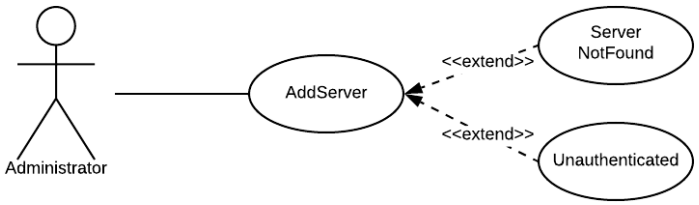


*Figure 5.2 - Use Case: Add Server to Permissions Manager*

**Use Case #3**

1. Name: Create New Role
2. Participating Actors: Administrator
3. Entry Conditions:

a. Administrator has been granted permission to use the application
b. Administrator is in the roles tab of the application

4. Exit Conditions:
    a. A new role is created and stored in the Configuration Manager database
5. Flow of Events:
    1. Administrator right clicks the roles overview table
    2. Administrator clicks “Add Role” button
    3. Table containing a list of stored procedures appears
    4. Administrator may filter procedures by server, database, or procedure name
    5. Administrator uses check boxes to indicate whether to grant or restrict access to certain stored procedures
    6. Administrator clicks save button
    7. Application establishes connection to Configuration Manager database
    8. Application inserts new role into Configuration Manager database
6. Alternate Flow of Events:
   [Permissions Manager Fails to Save Role]
    1. Display error message to notify the administrator that the role could not be saved

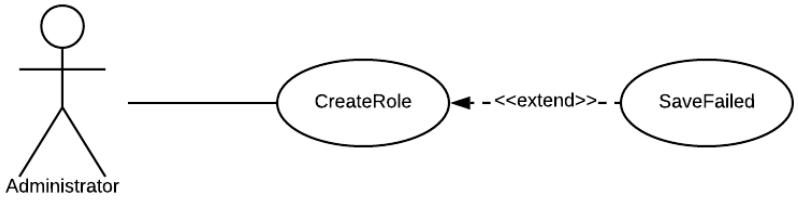


*Figure 5.3 - Use Case: Create New Role*

**Use Case #4**

1. Name: Update Role
2. Participating Actors: Administrator
3. Entry Conditions:
    a. Administrator has been granted permission to use the application
    b. Administrator is in the roles tab of the application
4. Exit Conditions:
    a. A role is stored in the Configuration Manager database with an updated procedure list
5. Flow of Events:
    1. Administrator right clicks a role in the roles overview table
    2. Administrator select “Edit Role” option from context menu
    3. able containing a list of stored procedures appears

4. Administrator may filter procedures by server, database, or procedure name
5. Administrator uses check boxes to indicate whether to grant or restrict access to certain stored procedures
6. Administrator clicks save button
7. Application establishes connection to Configuration Manager database
8. Application updates role in Configuration Manager database

6. Alternate Flow of Events:
   [Permissions Manager Fails to Update Role]
   1. Display error message to notify the administrator that the role could not be saved

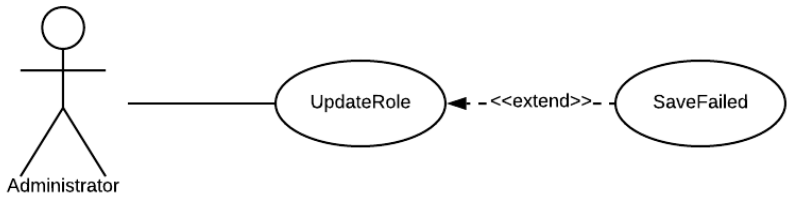


*Figure 5.4 - Use Case: Update Role*

**Use Case #5**

1. Name: Delete Role
2. Participating Actors: Administrator
3. Entry Conditions:
   a. Administrator has been granted permission to use the application
   b. Administrator is in the roles tab of the application
4. Exit Conditions:
   a. A role is deleted from the Configuration Manager database
5. Flow of Events:
   1. Administrator right clicks a role in the roles overview table
   2. Administrator selects “Delete Role” option from context menu
   3. Application establishes connection to Configuration Manager database
   4. Application remove role from Configuration Manager database
6. Alternate Flow of Events:
   [Permissions Manager Fails to Delete Role]
   1. Display error message to notify the administrator that the role could not be deleted

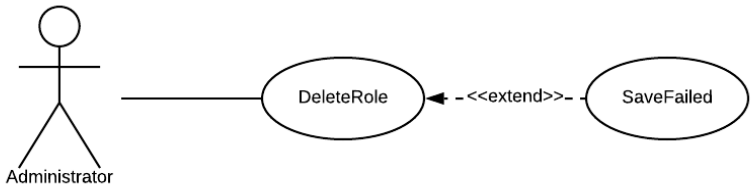


*Figure 5.5 - Use Case: Delete Role*

**Use Case #6**

1. Name: Update User's Roles
2. Participating Actors: Administrator
3. Entry Conditions:
    a. Administrator has been granted permission to use the application
    b. Administrator is in the users tab of the application
4. Exit Conditions:
    a. Roles are added to a user and that user is granted the permissions associated with the roles
5. Flow of Events:
    1. Administrator right clicks a user in the users overview table
    2. Administrator selects "Edit Roles" option from context menu
    3. Administrator may filter procedures by server, database, or procedure name
    4. Administrator uses check boxes to apply roles to user
    5. Administrator clicks save button
    6. Application establishes connection to Configuration Manager database
    7. Application updates user's roles in Configuration Manager database
6. Alternate Flow of Events:
   [Permissions Manager Fails to Add Roles to User]
    1. Display error message to notify the administrator that the user's roles could not be updated

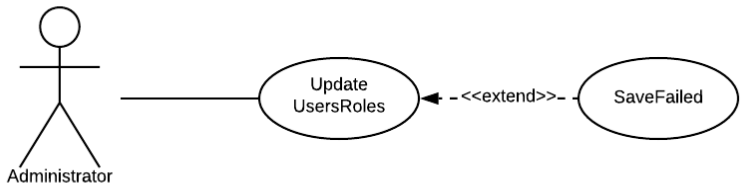


*Figure 5.6 - Use Case: Update User's Roles*

## 5.4 User Interface Mockups

Due to security and data privacy concerns, our team was not permitted to publish screenshots of the completed permissions manager. Nevertheless, the user mockups created during the preparation phase of this project, which served as the foundation for the final user interface, capture the essence of what each view contains.

When the permissions manager launches for the first time, the user will be prompted to add a new server to the application (Figure 5.7). The user will then need to provide a server address, port number, server type, and login credentials (Figure 5.8). If the permissions manager is successful in establishing a connection to the provided server, information about the database and stored procedures residing on that server will be extracted and displayed in the server directory pane of the permissions manager.

The tab bar contains icons that represent the three types of services that the application manages: database servers, user's roles, and the roles themselves. Each tab also contains two subviews, which alternate depending on the application state. When a tab is clicked, the user will initially be presented with the overview subview (Figure 5.9 and 5.11). As expected, the overview screen provides the user with a summary of the data that can be modified. The servers overview screen presents a list of servers that have been added to the application's configuration and exhibits the number of databases and stored procedures that reside within each server. The user's overview screen contains the set of users in the provisioned Active Directory group along with the roles that each user has been assigned (Figure 5.9). Similarly, the roles overview screen itemizes all the roles in the Configuration Manager database and maps them to the stored procedures that have been authorized for each one (Figure 5.11).

The second subview of each tab presents some form of an editor that can be used to modify the data in the application's data stores. The servers editor enables the addition of new server connections (Figure 5.8), the users editor provides a means of adding roles to users (Figure 5.10), and the roles editor allows for stored procedures to be added and removed from a role (Figure 5.12). Overall, the final user interface is structured similarly to the original mockups, however, the user interface was tweaked over several iterations to provide users with an intuitive and productive experience.

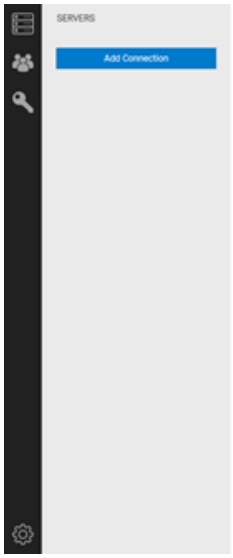


*Figure 5.7 - Initial Launch Screen*

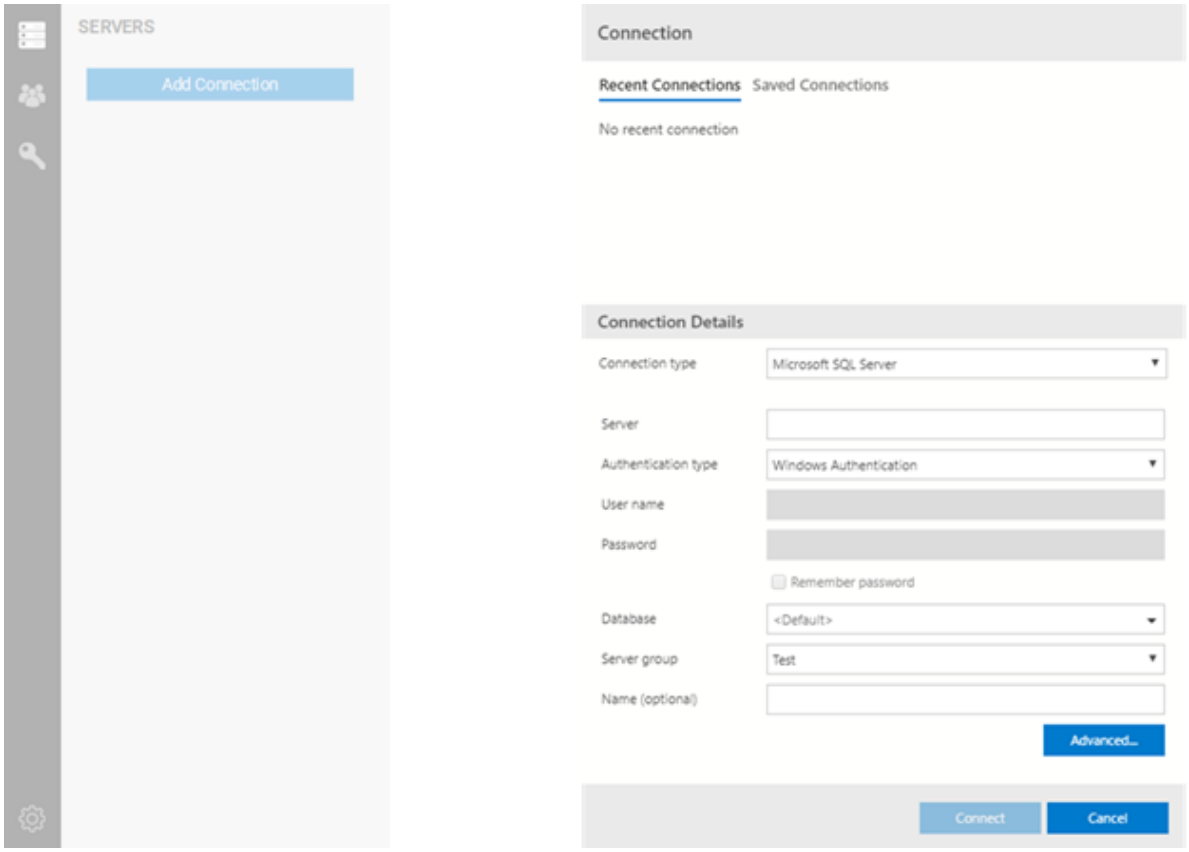


*Figure 5.8 - Adding Server Connections*

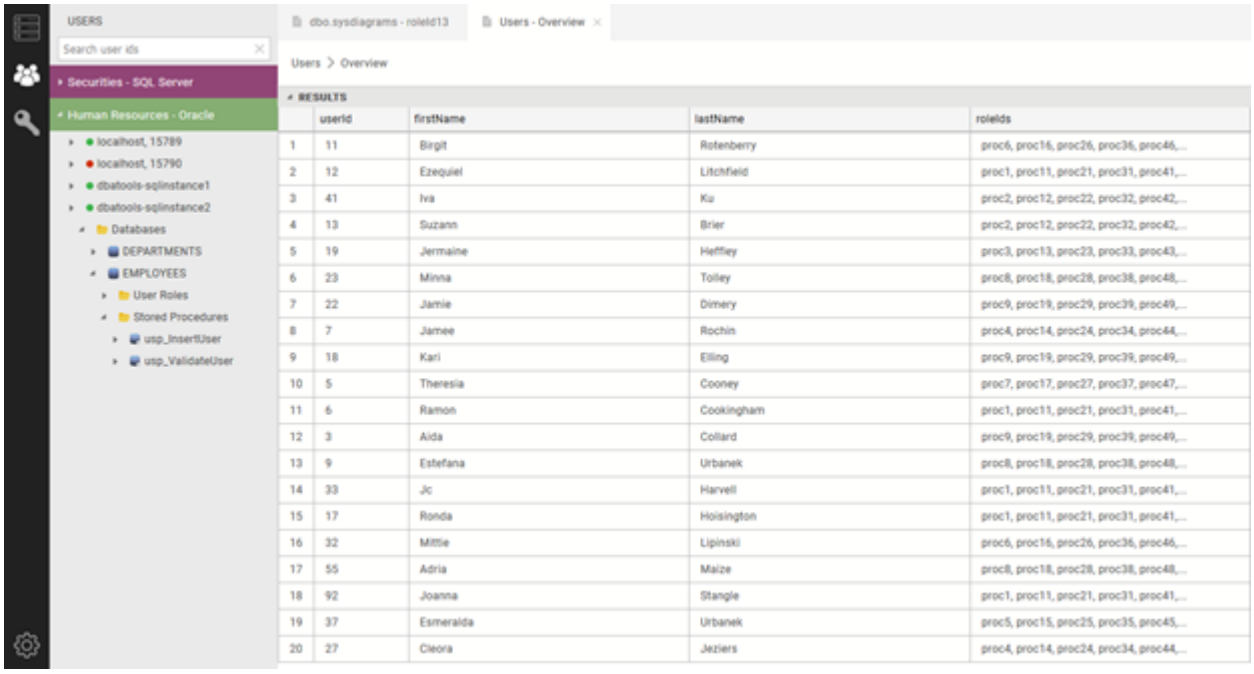


*Figure 5.9 - Users Tab: Overview*

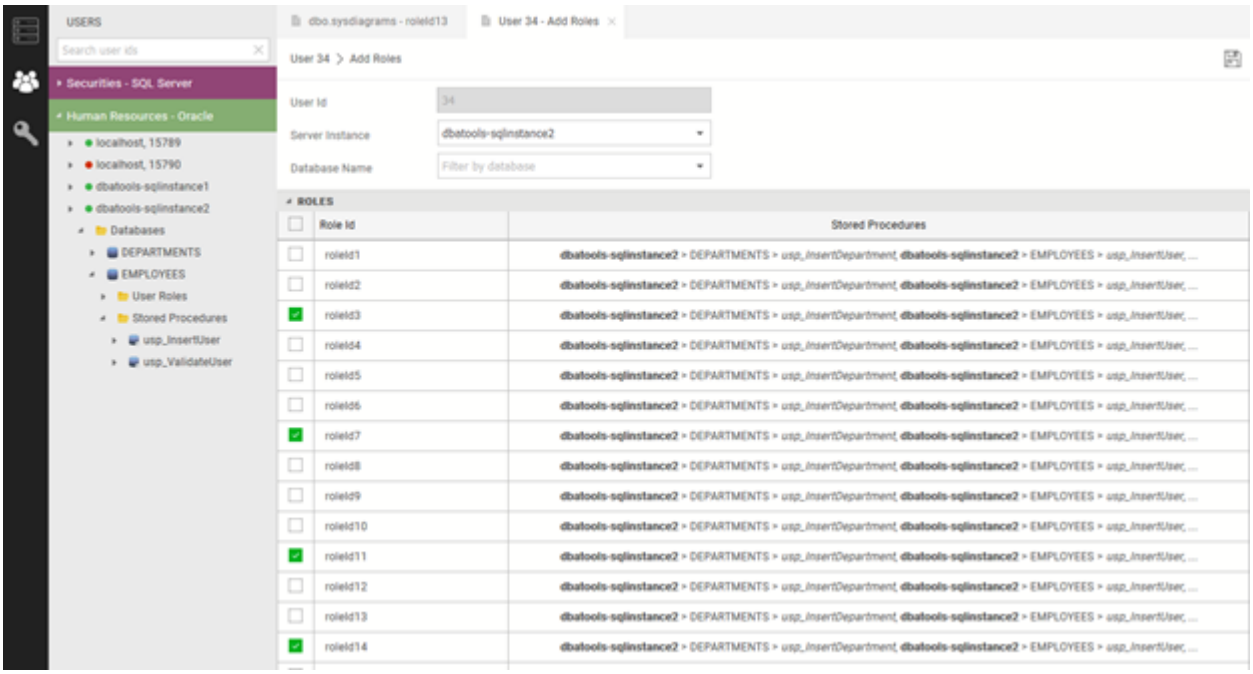


*Figure 5.10 - Users Tab: Editor*

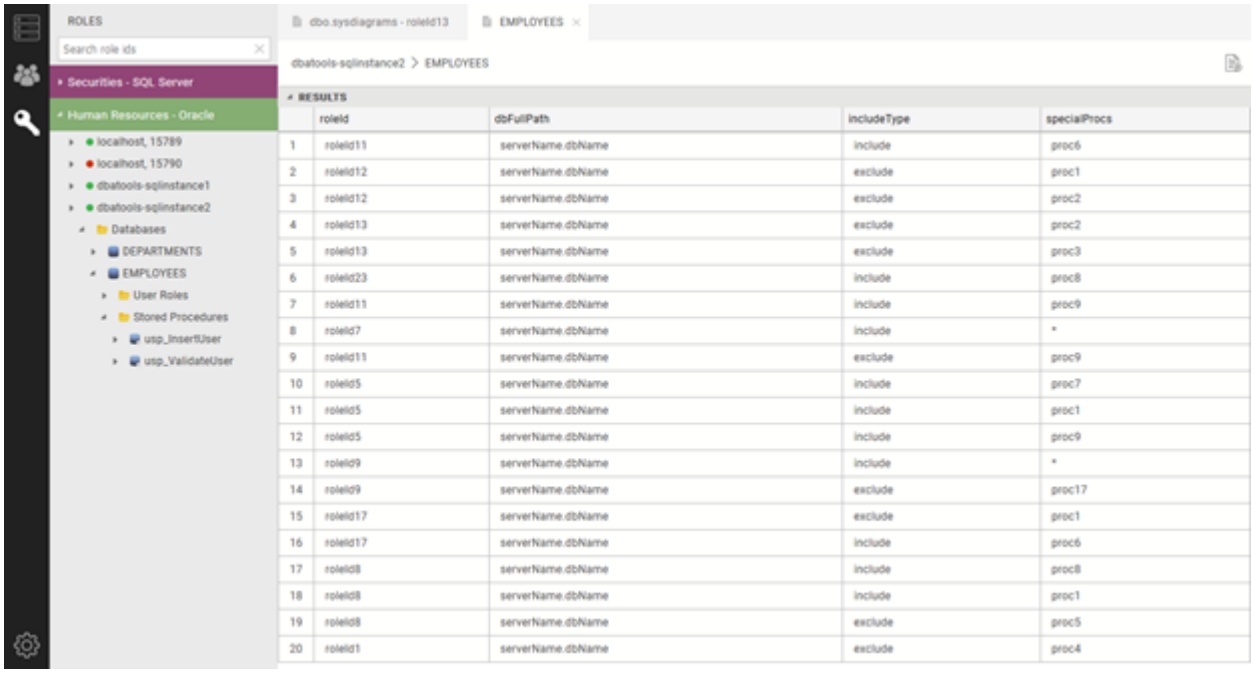


*Figure 5.11 - Roles Tab: Overview*

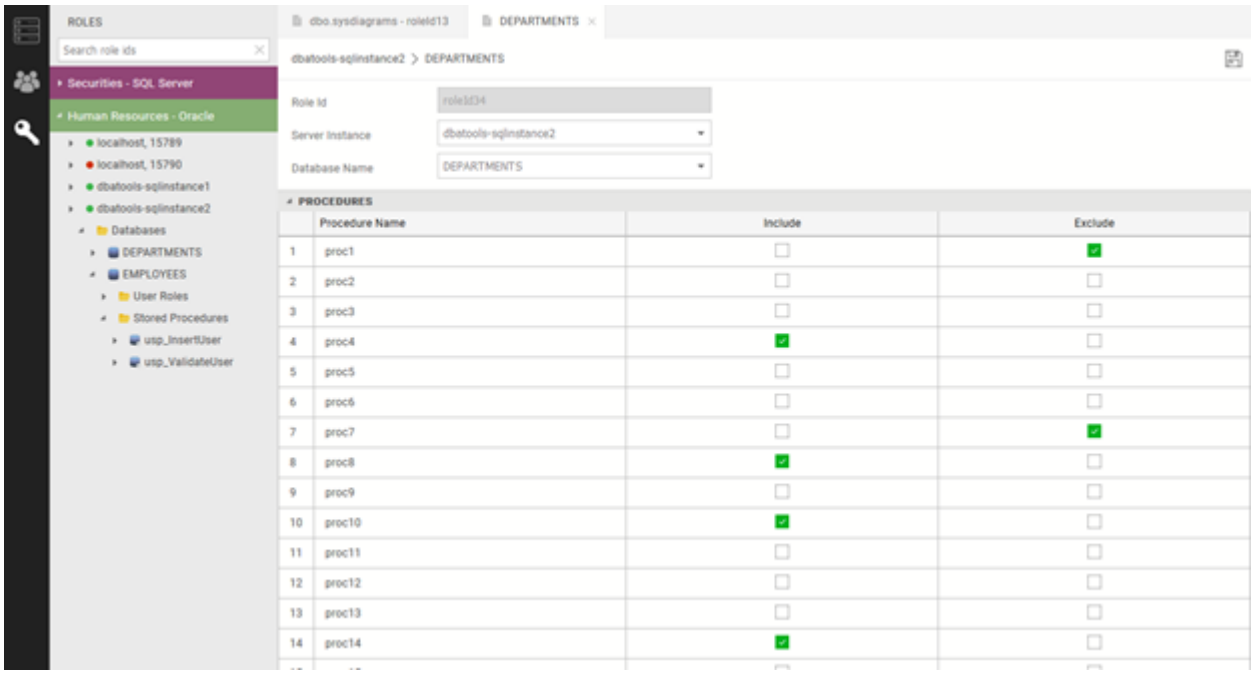


*Figure 5.12 - Roles Tab: Editor*

# Chapter 6: Product Design

## 6.1 Current Access Model

BNP Paribas' client applications assume the role of the presentation layer in a two-tier architecture (see Figure 6.1). The second constituent of the current access model is the set of databases that the applications rely on to present users with relevant information. The company primarily stores its data in Microsoft SQL Server and Oracle databases, each of which have their own authentication processes. Windows Authentication can be used in conjunction with SQL Server databases to abstract authentication from the user, however, Oracle databases require credentials to be hard-coded in the client or manually entered by the user. This practice of authenticating with user entered credentials is both tedious and insecure. Even if every user has a unique set of credentials, individuals can share credentials and enable access to and modification of confidential data. Consequently, BNP Paribas sought out a solution to this security vulnerability and adopted company policies that prohibit the access of databases directly from client applications.

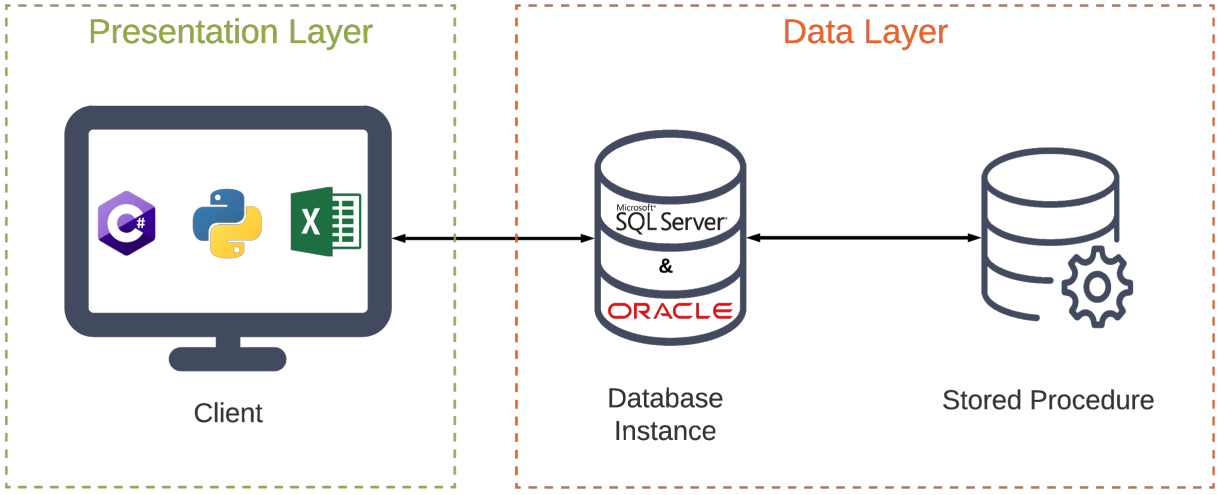


*Figure 6.1 - BNP Paribas' Database Access Model*

## 6.2 Architectural Design

### 6.2.1 Middleware Architecture

As mentioned in Section 2.2.1, a two-tier architecture works well for small, rarely changing applications, but the need for a low-maintenance, highly-scalable, centralized authentication service suggested the addition of an application layer to BNP's access model. The security, scalability, and flexibility improvements of a three-tier architecture sufficiently justified the loss in performance associated with the conversion from a two-tier system.

Figure 6.2 depicts our augmentation of BNP Paribas' access model, which incorporates a centralized authentication service that serves as the medium of communication between client applications and company databases. The main functionality of the database middleware includes authenticating users, authorizing users, and interacting with databases to process and relay data returned from stored procedures.

Instead of storing credentials in the client or relying on users to manually enter them, this improved model leverages the NT LAN Manager protocol to abstract the authentication process, regardless of the type of database. If the client is configured to use Windows Authentication, all a user must do is log into a company workstation and the client will automatically transmit the hash of the user's credentials to the middleware. The middleware then validates the credentials with BNP's Active Directory domain controller (see Figure 6.7 for authentication details). If the authentication is successful, the middleware will search through BNP's internal Configuration Manager to ensure that the user is permitted to execute the stored procedure that they requested. If both the authorization and authentication checks succeed, the middleware will connect to the specified database and execute the appropriate stored procedure. The returned data is then parsed into JSON and sent back to the client. It is important to note that all communications between the clients and middleware are done through HTTPS and JSON. Failure at any point in the aforementioned process will result in an HTTP response with an applicable status code and a descriptive message.

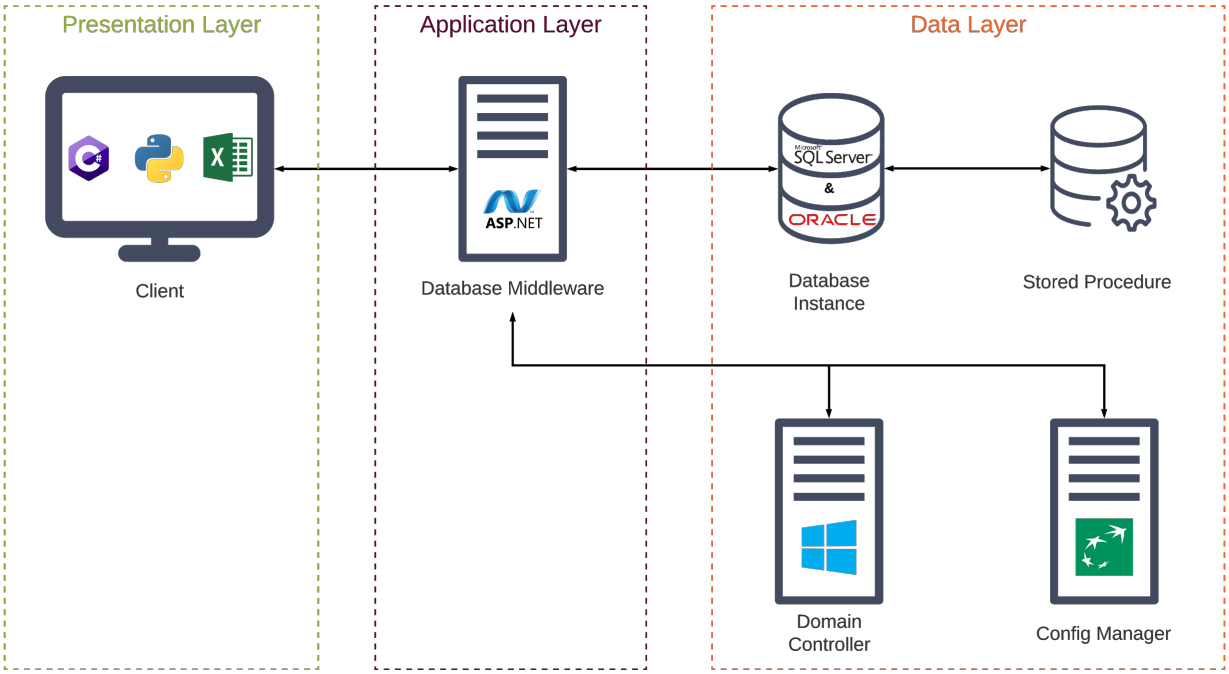


*Figure 6.2 - Database Middleware System Architecture*

### 6.2.2 Permissions Manager Architecture

The permissions manager does not communicate directly with the database middleware but interacts with many of the same data reservoirs that the middleware does (see Figure 6.3). The permissions manager interacts with BNP's Active Directory domain controller to obtain a list of users in the Active Directory group provisioned for this solution. The permissions manager also interacts with the same set of databases as the middleware to supply administrators with an auto-populated list of procedures that can be added to a role. Administrators can then use the permissions manager to create and assign roles that contain a set of authorized stored procedures. Information pertaining to roles and their assignments is retrieved and stored in the Configuration Manager. Consequently, when a system administrator uses the permissions manager to alter Configuration Manager data, he/she is effectively managing the authorization process mentioned in the previous section. To provide administrators with an accessible and responsive means of managing database access, all this information is presented via a user-friendly ReactJS.NET web application.

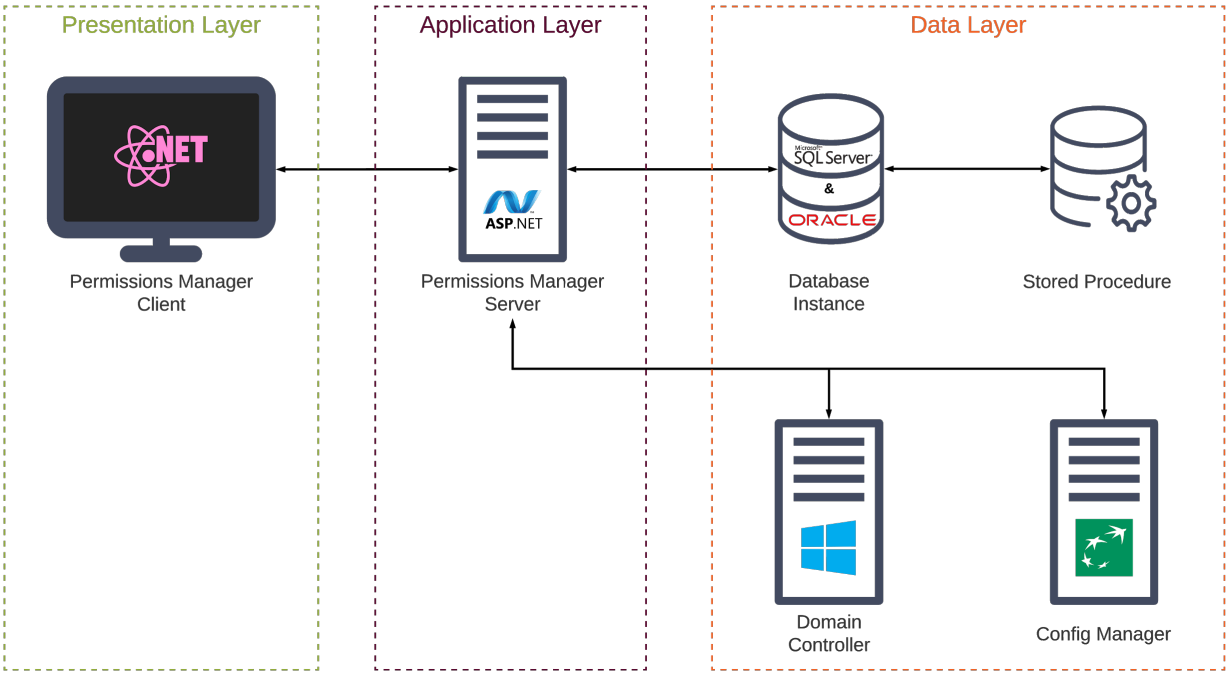


*Figure 6.3 - Permissions Manager System Architecture*

Since all of BNP Paribas' data resides in internally hosted databases, leveraging a third-party hosting service, such as Amazon Web Services, not only would require the company to loosen security protocols, but also would have been impractical in the allotted project timeframe. For them to function as intended, the middleware and permissions manager require access to BNP's internal databases. The connectivity between internal and external servers would have entailed opening flows from the company's protected production environment to an external hosting server. Furthermore, the permissions manager reads from and writes to BNP's Configuration Manager database. Although the exact implementation is unknown to us, it is not unreasonable to assume that securing the Configuration Manager for external exposure would be a substantial project on its own. Overall, the decision between hosting and self-hosting for our solution was not a difficult one. Choosing to self-host the middleware and permission manager using IIS ensured that developers reserve the ability to tweak server settings and that flows to BNP's databases remained exclusively internal.

### 6.2.3 Class Diagrams

The middleware service consists of eight classes (see Figure 6.4). When a client application sends a request to our middleware solution, it stores the parameters from the request body into the fields in the Credentials class. The RequestsController is responsible for checking that all the necessary fields are present and instantiating either a SQLServerControllerImpl or an OracleDBControllerImpl depending on the fields. The middleware adapts the factory pattern to instantiate the corresponding database controller object. Therefore, our Middleware does not have to specify the exact class of object that will be created.

To implement this pattern, we had an interface, IDatabaseConnection, and had the two child classes SQLServerControllerImpl and OracleDBControllerImpl implement this interface. The Middleware calls the build function in the DatabaseFactory class to build the corresponding database controller object based on the serverType in the Credentials. Once either the SQLServerControllerImpl or OracleDBControllerImpl is built, the controllers are responsible for handling the specific flow of operations needed to execute the stored procedure that the user requested. After the user is authenticated through NTLM, the IdentityController extracts the requestor's username. The AuthorizationController then uses the username retrieved from the IdentityController to check BNP's Configuration Manager application to see whether the user has permission to access the specified stored procedure. If the user has authorization to access, the middleware will attempt to execute the stored procedure.

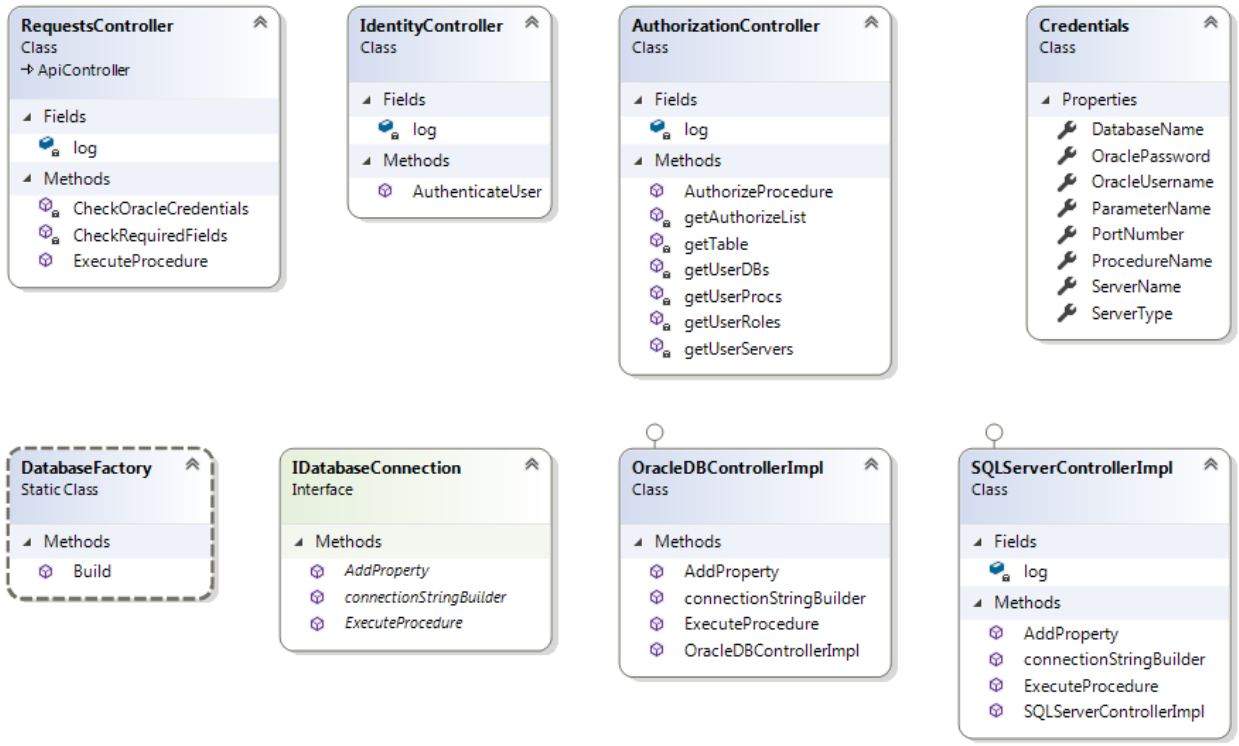


*Figure 6.4 - Middleware Service Class Diagram*

To ensure that the codebase for the permissions manager remained modular and readable, its vast functionality was divided into 25 classes (see Figure 6.5). The first two rows of classes represent those that were used to interface with BNP Paribas' Configuration Manager database. For each of the tables in the Configuration Manager that the permissions manager had to modify (servers, databases, stored procedures, roles, and authorization), there exists a controller that manages the functionality for querying the table, adding entries to the table, and removing entries from the table. There is also a utility class that provides methods for altering DataTable entries and for retrieving tables from the Configuration Manager. Likewise, the SQLServerController and OracleSeverController are responsible for interfacing database instances of their respective types. These two controllers implement the IDBServerController interface, which defines the functionality that future server controllers must exhibit. For each database server that the permissions manager must interface with, the DBServerDelegator uses the factory pattern to create an appropriate server controller.

The HomeController manages all the routes for the permissions manager and the ActiveDirectoryController interfaces and extracts user data from the provisioned Active

Directory group. Every day at 8 PM, the permissions manager removes all entries in the authorization table that pertain to users that are no longer in the Active Directory group. This process is managed by the JobScheduler, which executes the MaintenanceJob that calls the CleanUserData method of the MaintenanceController. The remaining 9 data model classes are employed to define structure and facilitate data access. Specifically, the CMServer, CMDatabase, and CMProcedure models define templates that the Configuration Manager controllers use to validate table entries. The 6 data model classes named without the CM prefix represent the structure of servers, databases, procedures, roles, users, and server connections that the permissions manager client uses to render results to a user. Because the permissions manager client has several data filtering mechanisms, these models, as opposed to the CM models, are optimized for search. The class diagram for the permissions manager may appear unwieldy at first, but this modularity should allow future developers to quickly find what they are looking for and to easily make changes to the code.

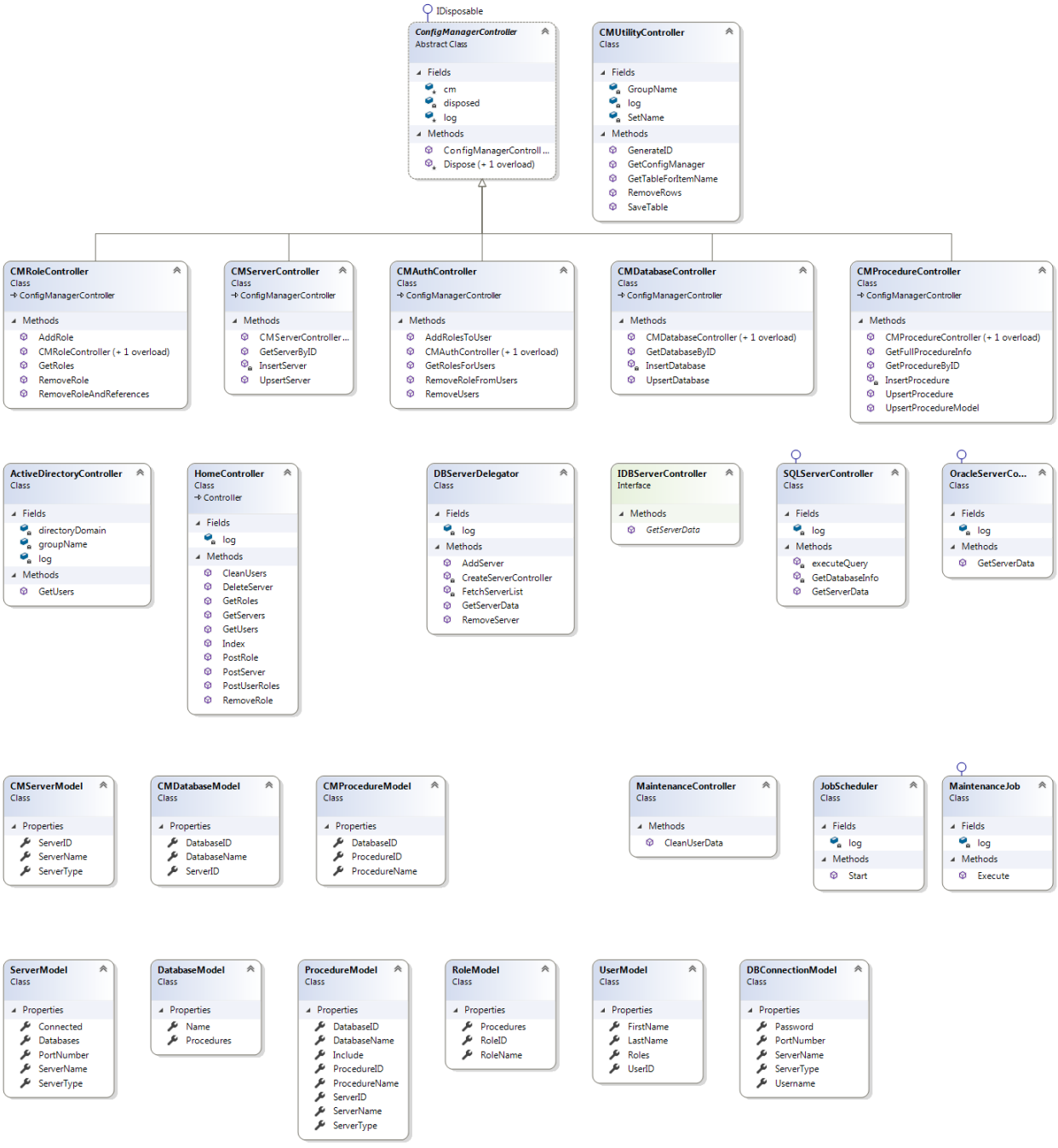


*Figure 6.5 - Permissions Manager Class Diagram*

### 6.2.4 Entity Relationship Diagrams

The procedures that the middleware executes are stored on Oracle and Microsoft SQL Server databases that existed prior to the onset of this project. The middleware and permissions managers are designed to handle an undefined number of servers and therefore databases.

Consequently, this section focuses on tables in the Configuration Manager database, which store the data that specifically pertains to the functionality of our solution. The Configuration Manager database refers to BNP's custom data repository used to house information for applications throughout the company. The database structure is quite similar to a flat-file database, in that is has very little structure and does not recognize relationships between records. Due to its lack of constraints, entries in any of the tables can be manually modified as desired. To provide some layer of protection against corruption and inconsistency, the schema was normalized as much as possible (see Figure 6.6). The tables are also loosely coupled so that if a server gets moved or renamed, because other tables rely on static IDs, only one entry in the server table needs to be modified.

Since the main use of the Configuration Manager is to store information for authorizing a user, the role, authorization, and procedure tables are the most substantive. The database and server tables are merely there to identify the database and server that a procedure belongs to. When the middleware authorizes a user, the authorization table is traversed to determine the roles that belong to the authenticated user. Using that role set, the middleware then tries to match the procedure information supplied by the client with the procedure information corresponding to each of the user's role. Since the middleware uses this Configuration Manager database for looking up information about a user, its access is restricted to read-only. The permissions manager, on the other hand, is a tool expressly developed for maintaining the database. When an administrator creates a new role, the permissions manager will create entries in the Procedure, Database, and Server tables, which are used to uniquely identify the procedures in each role. Modifying a user's role set is simpler because only the Authorization table is impacted. It is important to note that to save space and make the database more manageable, procedure data is inserted on an as-needed basis, as opposed to importing data from all the procedures on all databases when a new server is added to the application.

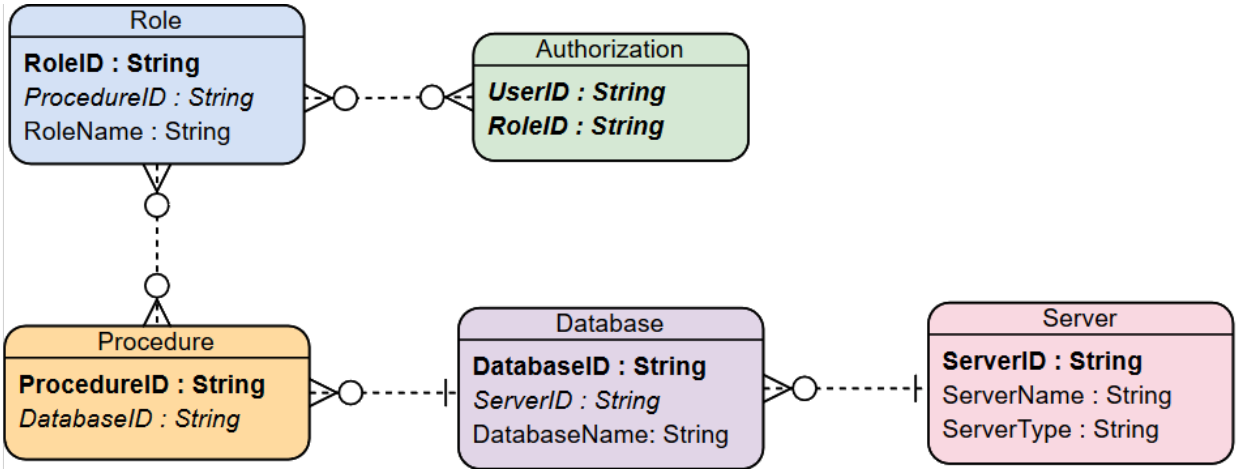


*Figure 6.6 - Configuration Manager Database Entity Relationship Diagram*

### 6.2.5 Sequence Diagrams

**NTLM Authentication**

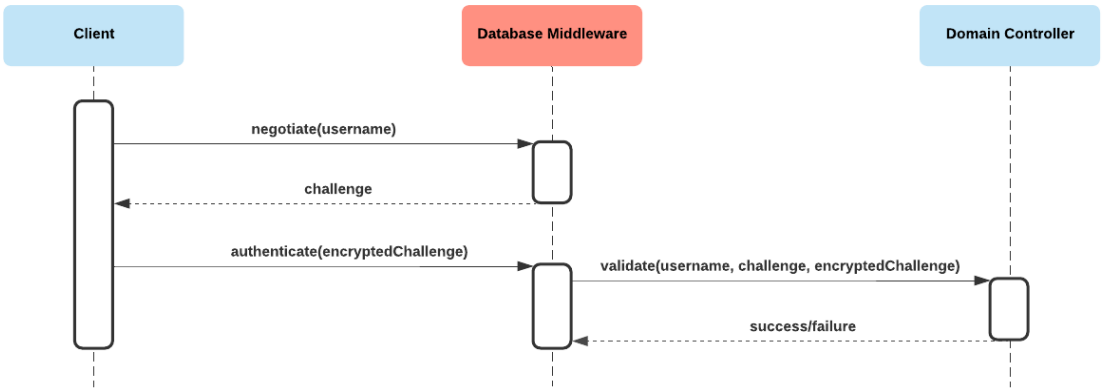


*Figure 6.7 - Sequence Diagram: NTLM Authentication*

1. A client sends a username in plaintext to the middleware
2. The middleware generates a 16-byte random number (challenge) and sends it to the client
3. The client encrypts the challenge with the hash of the user's password and sends the encrypted challenge to middleware
4. The middleware sends the username, challenge, and encrypted challenge to the Active Directory domain controller

5. The domain controller uses the username to retrieve the hash of the user's password and encrypts the challenge. If the encrypted challenge is identical to the one computed by the domain controller, the client is successfully authenticated.

## Middleware

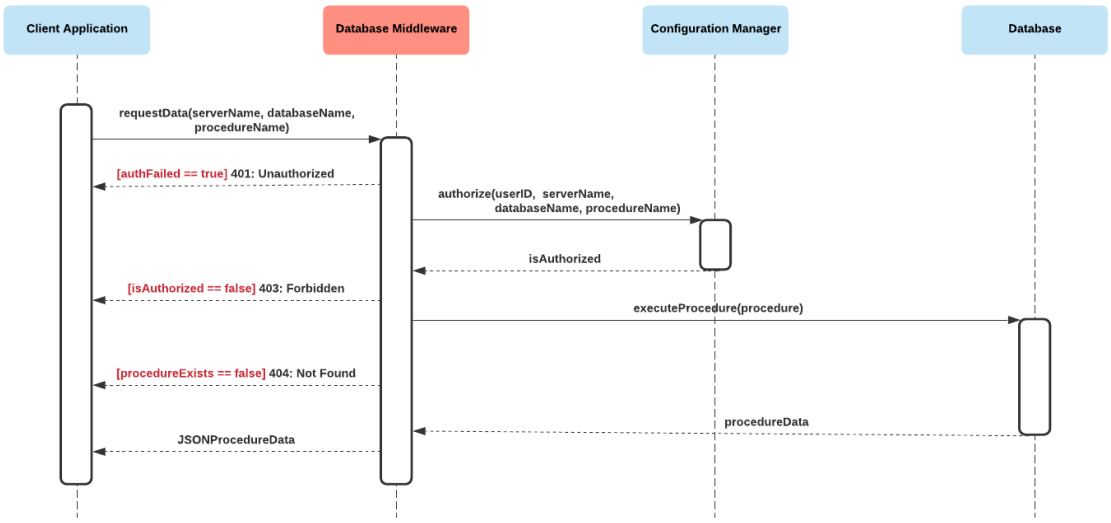


*Figure 6.8 - Sequence Diagram: Middleware*

1. A client sends the middleware the name of a procedure to execute with the database and server that the procedure belongs to
2. The middleware authenticates the client using NTLM
3. The middleware sends the client's identity with the stored procedure information to the Configuration Manager database
4. The Configuration Manager database determines if the client is authorized to execute the specified stored procedure and notifies the middleware
5. The middleware executes the stored procedure that the client requested
6. The middleware forwards the data to the client in JSON object array

**Permissions Manager – Initialization**

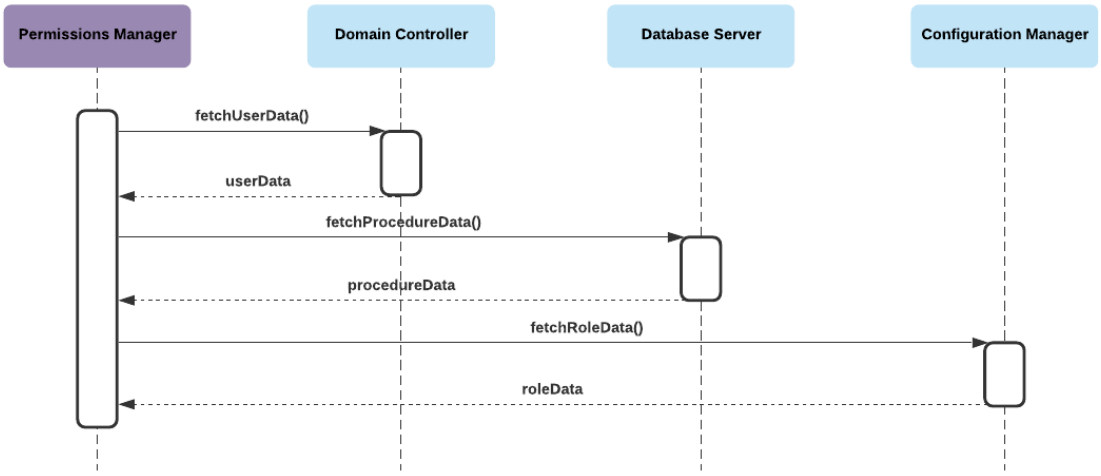


*Figure 6.9 - Sequence Diagram - Permissions Manager: Initialization*

1. The permissions manager makes a request to the Active Directory domain controller to get a list of users in the provisioned active directory group
2. The domain controller responds with data pertaining to users in Active Directory group
3. The permissions manager makes a request to the stored database servers to get data about the procedures
4. The database server responds with a collection of databases and the procedures each one has
5. The permissions manager makes a request to Configuration Manager database to get role data
6. The Configuration Manager database responds with a set of roles and a mapping between users and their roles

**Permissions Manager - Add Server**

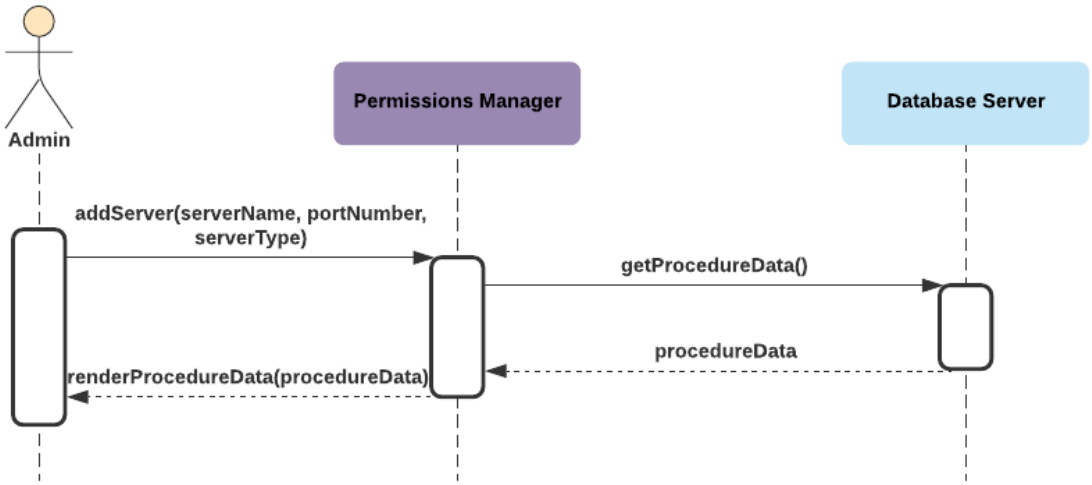


*Figure 6.10 - Sequence Diagram - Permissions Manager: Add Server*

1. An administrator provides the permissions manager with the name, port number, and server type of the database server to add to the service
2. The permissions manager fetches database and stored procedure data from the corresponding database server
3. The database server returns the database and stored procedure data to the permissions manager
4. The permissions manager rerenders the web page to reflect the updated data

**Permissions Manager - Add Role**

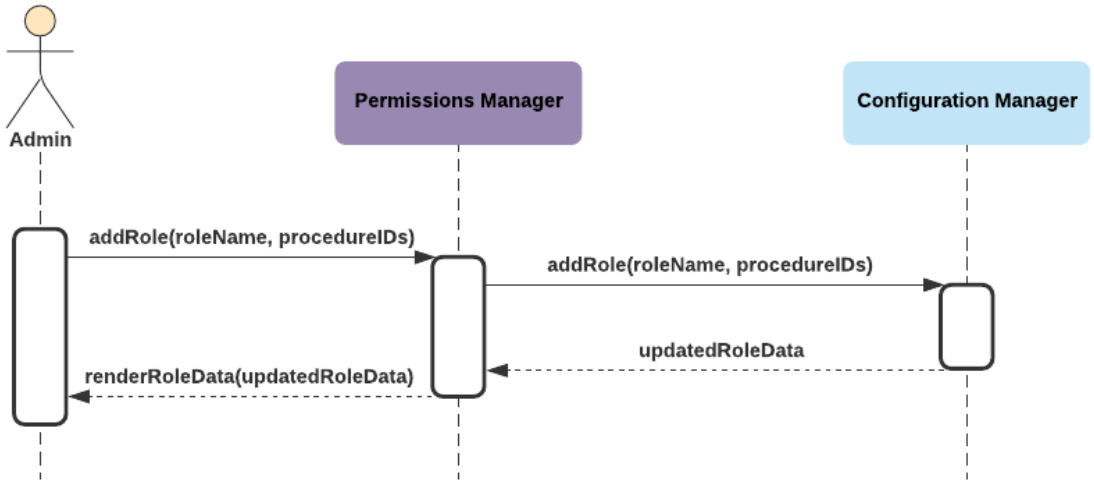


*Figure 6.11 - Sequence Diagram - Permissions Manager: Add Role*

1. An administrator uses the permissions manager to assign a role name to a set of stored procedure IDs
2. The permissions manager adds rows in the appropriate tables of the Configuration Manager database to create a new role and assign the procedures to it
3. The Configuration Manager returns the updated set of all roles to the permissions manager
4. The permissions manager rerenders the client to reflect the updated data

**Permissions Manager - Remove Role**

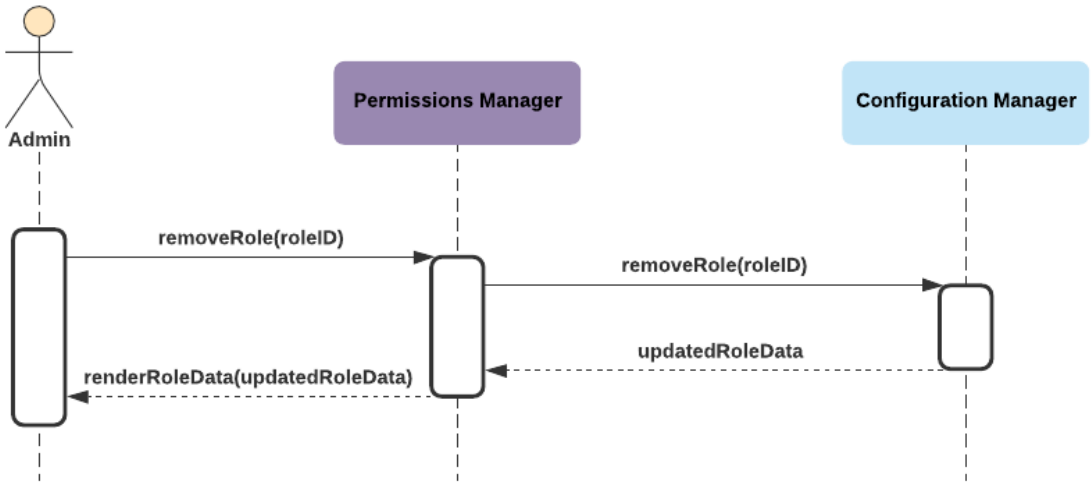


*Figure 6.12 - Sequence Diagram - Permissions Manager: Remove Role*

1. An administrator presses the delete button on a role in the permissions manager
2. The permissions manager removes all entries in the Configuration Manager database that pertain to the role
3. The Configuration Manager database returns the updated set of all roles to the permissions manager
4. The permissions manager rerenders the client to reflect the updated data

**Permissions Manager - Update User's Roles**

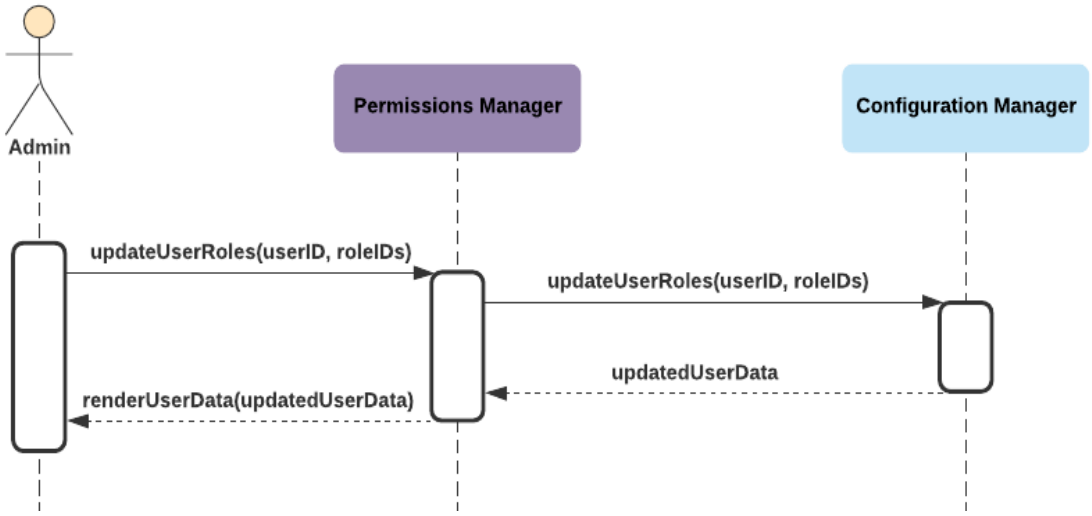


*Figure 6.13 - Sequence Diagram - Permissions Manager: Update User's Roles*

1. An administrator uses the permissions manager to modify the roles that a user has
2. The permissions manager makes a request to Configuration Manager database to update the set of role IDs that correspond to the user
3. The Configuration Manager database returns the updated mapping of users to roles to the permissions manager
4. The permissions manager rerenders the client to reflect the updated data

# Chapter 7: Software Development

Our team scheduled weekly sprint planning meetings on Monday mornings, daily Scrum meetings, and retrospective/review meetings on Friday afternoons. The sprint planning meetings were used to set goals for the week and to determine which user stories we wanted to finish during the sprint. The daily scrum meetings were for us to report to each other what we had achieved the previous work day and what we aimed to get done during the present work day. The retrospective/review meetings enabled us to reflect on how the week went and how much we achieved for the sprint.

## 7.1 Sprint 1: October 10 - October 25

Since Monday was our first day arriving at BNP Paribas in New York, we scheduled all of our Agile meetings for the rest of the term after we got used to the work environment. Although we did not hold a formal Sprint Planning meeting on our first day, we had our user stories documented in our Trello board and discussed before the term started what we wanted to try to achieve by the end of the first week.

Table 7.1 below shows the Trello tasks and the user stories that the tasks pertained to for the first sprint. Red rows are Trello tasks that were not previously in our backlog and were added in during the sprint.

| **Trello Card Subject(s)** | **Description** | **Point** | **Related User Story/Stories** |
|---|---|---|---|
| Middleware | Restful API<br>- Receive POST requests from API<br>- Retrieve and parse data<br>- Send data back to client applications | 1 | As a manager, I want to prevent my employees from directly accessing company databases so that I can ensure the data has not been manually modified. |
| Middleware | Users + Roles tables<br>- Create database | 2 | As a manager, I want to restrict my employees' access to databases so that they cannot view or manipulate data |

| | | | |
|---|---|---|---|
| | tables in Configuration Manager | | without explicit authorization. |
| Middleware | Create client applications | 1 | As an engineer, I want a solution that is compatible with applications written in many programming languages so that I do not have to create a new service for each type of application. |
| Middleware; Documentation | API Documents | 1 | As an employee, I want to access company databases quickly so that I have the information I need to make informed decisions |
| Permissions Manager | Basic GUI | 1 | As an administrator, I want to be able to access the GUI without having to install an application on my computer. |
| Permissions Manager | Controllers | 3 | As an administrator, I want user lists to auto-populate so that I do not have to manually enter information.<br><br>As an administrator, I want database lists to auto-populate so that I do not have to manually enter information.<br><br>As an administrator, I want stored procedure lists to auto-populate so that I do not have to manually enter information. |

*Table 7.1 - Sprint 1 Tasks*

For the first sprint, we encountered a few roadblocks that emanated from lack of work permissions. As we got used to the work environment, we found that most applications we needed to utilize for the term required us to fill out company requests to gain access to those applications. In the meantime, we completed tasks that did not require access to the applications we were waiting for. We templated a basic Middleware API and created client applications in C#, Python, and VBA code to send, receive, and process data from our Middleware API. In parallel, we also started creating the Permissions Manager UI. We discovered that the company

almost exclusively uses Internet Explorer. This unfortunately rendered contemporary features of our UI codebase unusable. Consequently, a good portion of the week was dedicated to ensuring the UI was compatible with Internet Explorer 11.

During our retrospective/review meeting for this sprint, we reflected on having to wait for certain work permissions to be granted before we could start on more complex tasks. Since we did not have access to the Configuration Manager application, we could not create the database tables for one of our Trello tasks for the sprint. Despite these roadblocks, we felt we achieved progress for the first sprint. Below is a burndown chart (Figure 7.1) comparing the number of points before we began the first sprint and the number of points remaining after the sprint. Since we added a new task, we still had a good number of points remaining.

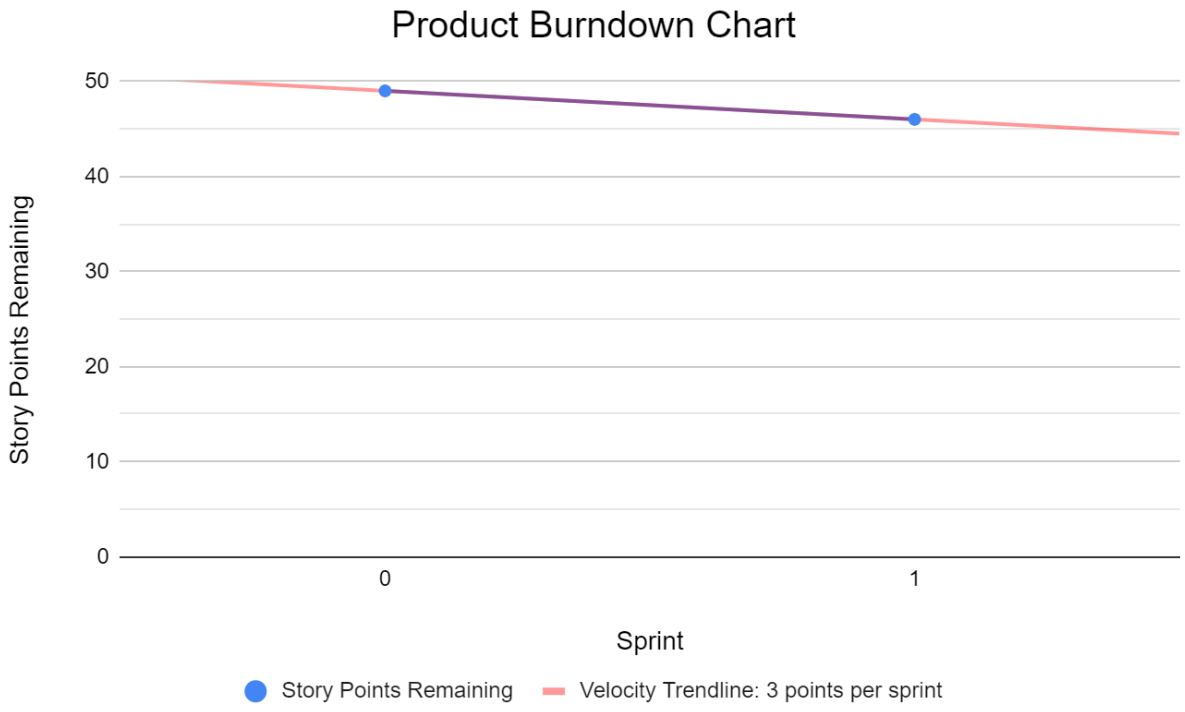


*Figure 7.1 - Sprint 1 Burndown Chart*

## 7.2 Sprint 2: October 28 - November 1

By the second sprint, our team had a good grasp of using the Agile methodology during our project. One of our main goals in our second sprint planning meeting was to get any code we had onto a formal Bitbucket repository, as we were delayed the first week from the lack of

permissions. We also aimed to pull data from the company's databases and stored procedures in our middleware API and permissions manager UI.

Table 7.2 below shows the Trello tasks and the user stories that the tasks pertained to for the second sprint. Red rows are Trello tasks that were not previously in our backlog and were added in during the sprint. Yellow rows are Trello tasks that rolled over from previous sprints if they still needed work.

| **Trello Card Subject(s)** | **Description** | **Point** | **Related User Story/Stories** |
|---|---|---|---|
| Middleware | Authentication System | 13 | As a manager, I want to restrict my employees' access to databases so that they cannot view or manipulate data without explicit authorization. |
| Middleware | Authorization | 3 | As a manager, I want to restrict my employees' access to databases so that they cannot view or manipulate data without explicit authorization. |
| Middleware | Execute stored procedures - connect to server and databases | 2 | As an employee, I want to access company databases quickly so that I have the information I need to make informed decisions. |
| Middleware | Execute stored procedures - being able to execute | 2 | As an employee, I want to access company databases quickly so that I have the information I need to make informed decisions. |
| Middleware | Users + Roles tables<br>- Create database tables in Configuration Manager | 2 | As a manager, I want to restrict my employees' access to databases so that they cannot view or manipulate data without explicit authorization. |
| Permissions Manager | Pull database and stored procedure data | 3 | As an administrator, I want database lists to auto-populate so that I do not have to manually enter information.<br><br>As an administrator, I want stored procedure lists to auto-populate so |

| | | | |
|---|---|---|---|
| | | | that I do not have to manually enter information. |
| Permissions Manager | Pull from active directory | 8 | As an administrator, I want user lists to auto-populate so that I do not have to manually enter information. |
| Permissions Manager | Controllers | 3 | As an administrator, I want user lists to auto-populate so that I do not have to manually enter information.<br><br>As an administrator, I want database lists to auto-populate so that I do not have to manually enter information.<br><br>As an administrator, I want stored procedure lists to auto-populate so that I do not have to manually enter information. |

*Table 7.2 - Sprint 2 Tasks*

For the second sprint, we were able to successfully connect to databases and execute stored procedures from our middleware API and display database information in the permissions manager. We were also able to successfully pull the list of users from the company's Active Directory group. By the end of the second week, we were able to push all of our code to a Bitbucket repository so that we could individually access the same code from our workstation computers.

Some obstacles that we encountered during our second sprint included trying to implement an authentication system. BNP Paribas requested us to use Kerberos authentication for our middleware. However, from doing online research and clarifying some information with the company's Single Sign-On team, we learned that trying to implement Kerberos authentication with the servers that we needed to access would take months. In the meantime, our supervisor and representatives from the Single Sign-On needed to take some time to determine to either request servers for us so that we could implement Kerberos authentication, or use a different authentication protocol such as NT LAN Manager (NTLM). Another challenge that we faced was trying to create database tables in the company's Configuration Manager

application. We did not have direct access to the Configuration Manager application, but we tried to utilize the Configuration Manager NuGet package that BNP Paribas had. Since the Configuration Manager package required specific configuration settings in Visual Studio and we were unaware of alternatives at the time, running our applications with the Configuration Manager NuGet package was extremely difficult.

In our retrospective/review meeting for our second sprint, we mainly reflected on the roadblocks we encountered regarding the authentication system and the Configuration Manager package. We attempted to brainstorm alternative authentication solutions if Kerberos authentication was not feasible. We also noted that despite the obstacles we encountered this week, we were still able to make progress with regards to executing stored procedures from our middleware API and pulling data from the active directory group to our permissions manager. Below is a burndown chart (Figure 7.2) comparing the number of points before we began the first sprint and the number of points remaining after each sprint so far.

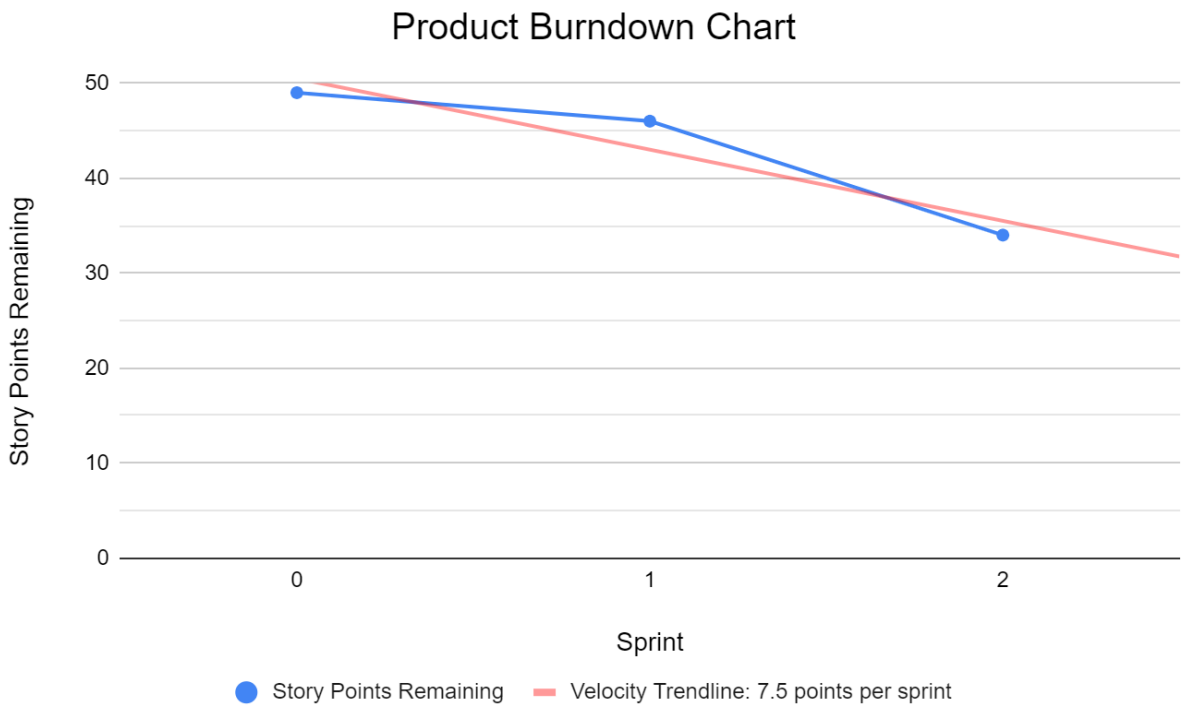


*Figure 7.2 - Sprint 2 Burndown Chart*

## 7.3 Sprint 3: November 4 - November 8

For our third sprint planning meeting, we wanted to prioritize getting our applications to run properly with the Configuration Manager NuGet package, since that was one of the roadblocks in our second sprint. We knew if we were able to resolve this obstacle, we would be able to start implementing the authorization for the middleware and functionality for editing the Configuration Manager from the permissions manager.

Table 7.3 below shows the Trello tasks and the user stories that the tasks pertained to for the third sprint. Red rows are Trello tasks that were not previously in our backlog and were added in during the sprint. Yellow rows are Trello tasks that rolled over from previous sprints if they still needed work.

| **Trello Card Subject(s)** | **Description** | **Point** | **Related User Story/Stories** |
|---|---|---|---|
| Middleware | Authentication System | 13 | As a manager, I want to restrict my employees' access to databases so that they cannot view or manipulate data without explicit authorization. |
| Middleware | Authorization | 3 | As a manager, I want to restrict my employees' access to databases so that they cannot view or manipulate data without explicit authorization. |
| Middleware | Users + Roles tables<br>- Create database tables in Configuration Manager | 2 | As a manager, I want to restrict my employees' access to databases so that they cannot view or manipulate data without explicit authorization. |
| Permissions Manager | Controllers | 3 | As an administrator, I want user lists to auto-populate so that I do not have to manually enter information.<br><br>As an administrator, I want database lists to auto-populate so that I do not have to manually enter information.<br><br>As an administrator, I want stored procedure lists to auto-populate so |

| | | | |
|---|---|---|---|
| | | | that I do not have to manually enter information. |
| Permissions Manager | Complete GUI | 5 | As an administrator, I want to edit multiple user permissions at once so that I can save time.<br><br>As an administrator, I want to limit manual input so that I make fewer mistakes. |
| Permissions Manager | Navigation Controller | 5 | As an administrator, I want to manage the stored procedures that a user can execute so that employees cannot view or manipulate data without explicit authorization. |
| Permissions Manager | Modify data in configuration manager | 3 | As an administrator, I want to manage the stored procedures that a user can execute so that employees cannot view or manipulate data without explicit authorization.<br><br>As an administrator, I want to edit multiple user permissions at once so that I can save time. |

*Table 7.3 - Sprint 3 Tasks*

Since we were still waiting for direct access to the Configuration Manager application, we requested that our supervisor create the database tables we needed by sending him an Excel template of what we wanted the tables to look like. After some help from other employees and our supervisor, we were also able to properly run our applications from Visual Studio with the Configuration Manager package, after setting environment PATH variables. Since we got this issue resolved at the beginning of our sprint, we focused on implementing authorization for our middleware and modifying the Configuration Manager tables from the permissions manager.

One obstacle that we ran into during this sprint was our supervisor needed us to fill out application forms to onboard our project into the company system and to request servers from the company's Single Sign-On team. The information needed for the forms was unclear and unintuitive, and there was no formal documentation that we could find to help us complete the

application forms. Our supervisor suggested that we clarify the information with representatives from the teams we needed to submit the forms to. Due to this, a portion of the sprint time was taken away from directly working on our project, and instead was dedicated to clarifying information and trying to complete the forms before continuing the following week.

During our retrospective/review meeting, we reflected on the significant progress we made on authorization and editing the Configuration Manager. After utilizing the code from the Configuration Manager package, we expressed dissatisfaction of the libraries that the package had to offer us. Nevertheless, we agreed to continue working with the package to the best of our abilities. We had to stay at work late several days during this sprint to determine the best ways to utilize their package libraries, so our applications would be as optimized as possible. We also addressed that some time needed to be allocated to work with people from other teams to fill out forms that related to components that our project needed. Below is a burndown chart (Figure 7.3) comparing the number of points before we began the first sprint and the number of points remaining after each sprint so far.

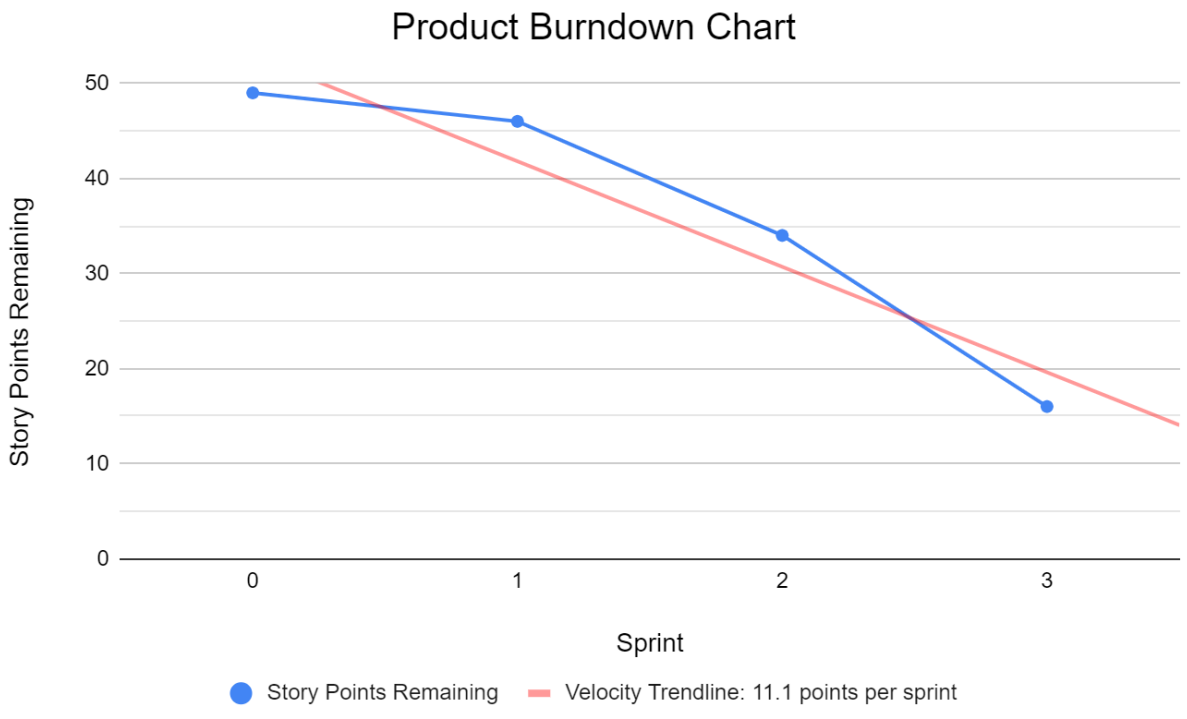


*Figure 7.3 - Sprint 3 Burndown Chart*

## 7.4 Sprint 4: November 11 - November 15

At our fourth sprint planning meeting, we needed to continue filling out the forms for our project. We were also scheduled to presented what we had accomplished so far to our supervisor, his line manager, and another BNP Paribas employee, so we wanted to focus on patching up any bugs and preparing for our initial demonstration. After we presented the solution, we received positive feedback and additional suggestions as to how we could further improve our current solution.

Table 7.4 below shows the Trello tasks and the user stories that the tasks pertained to for the fourth sprint. Red rows are Trello tasks that were not previously in our backlog and were added in during the sprint after our initial demonstration. Yellow rows are Trello tasks that rolled over from previous sprints if they still needed work.

| **Trello Card Subject(s)** | **Description** | **Point** | **Related User Story/Stories** |
|---|---|---|---|
| Middleware | Authentication System | 13 | As a manager, I want to restrict my employees' access to databases so that they cannot view or manipulate data without explicit authorization. |
| Middleware | Excel Plug in | 2 | As an engineer, I want a solution that is compatible with applications written in many programming languages so that I do not have to create a new service for each type of application. |
| Middleware; Permissions Manager | Deploy applications to remote server | 8 | As a manager, I want a solution that is hosted on a remote server so that employees from across the world can utilize its services.<br><br>As an administrator, I want to be able to access the GUI without having to install an application on my computer. |
| Permissions Manager | Feedback from demonstration<br>- Change layout display of | 2 | As an administrator, I want user lists to auto-populate so that I do not have to manually enter information. |

| | | | |
|---|---|---|---|
| | users<br>- Clean up of mismatched data every night | | |

*Table 7.4 - Sprint 4 Tasks*

This week, our supervisor provided us with remote servers to which we could begin deploying our project. One of our manager's colleagues to had prior experience deploying to remote servers and configuring NTLM authentication. Since we were still in the process of waiting for the company's Single Sign-On team regarding Kerberos authentication, we decided to temporarily configure NTLM authentication for our middleware API.

One major obstacle that we ran into during this sprint was the remote server's limited permissions. Before we could configure NTLM authentication, we needed to deploy our applications to the remote servers. However, we later found out that we could not automatically deploy to the remote servers from Visual Studio, and that we would have to manually copy over the locally deployed files into the remote server. During this process, we also discovered that the Configuration Manager package that brought about roadblocks during our second sprint produced more obstacles for this sprint. Unfortunately, our previous solution that worked on our local machines did not work for the remote server, so that became a problem that we needed to fix in the next sprint.

During our retrospective/review meeting, we reflected on our positive feedback from the presentation of our solution. We were able to successfully implement an Excel add-in that an employee requested. This would make receiving data from our Middleware API for the VBA clients more efficient and significantly easier for users to read. We also finished adding in the suggestions we received for the permissions manager UI. Namely, how certain information was displayed to users, and the clean-up job to remove old users from the authentication table of the Configuration Manager. We also addressed our concerns again with respect to the use of the Configuration Manager package, since this was the second sprint that we ran into problems with it. We aimed to resolve the obstacle in our next sprint and continue onward with the remaining tasks. Below is a burndown chart (Figure 7.4) comparing the number of points before we began the first sprint and the number of points remaining after each sprint so far.

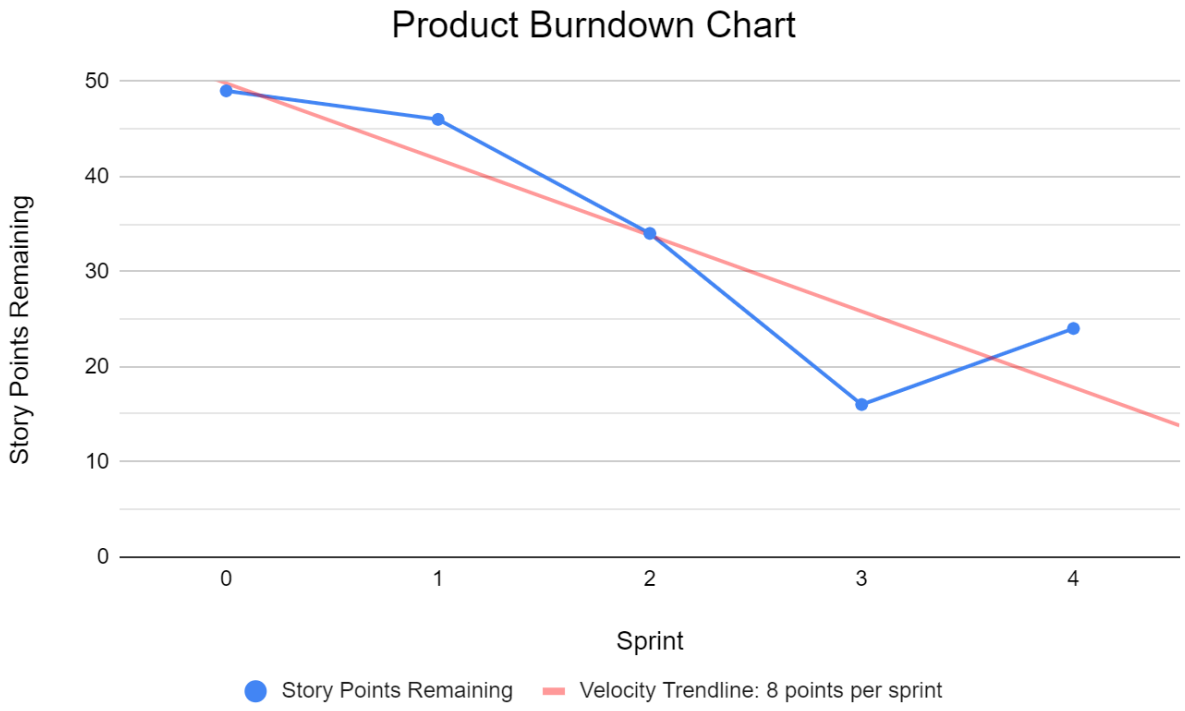


*Figure 7.4 - Sprint 4 Burndown Chart*

## 7.5 Sprint 5: November 18 - November 22

At our fifth sprint planning meeting, we aimed to resolve the error that we ran into during the previous sprint, regarding the deployment of our project to the remote server. Once we were able to solve that issue, implementing the authentication protocol would be our next priority. We were also scheduled to meet with our supervisor to discuss the diagrams we had created so far, so we planned to prepare for that meeting in addition to working on our tasks this sprint.

Table 7.5 below shows the Trello tasks and the user stories that the tasks pertained to for the fifth sprint. Red rows are Trello tasks that were not previously in our backlog and were added in during the sprint after our initial demonstration. Yellow rows are Trello tasks that rolled over from previous sprints if they still needed work.

| **Trello Card Subject(s)** | **Description** | **Point** | **Related User Story/Stories** |
|---|---|---|---|
| Middleware | Authentication System | 13 | As a manager, I want to restrict my employees' access to databases so that |

| | | | |
|---|---|---|---|
| | | | they cannot view or manipulate data without explicit authorization. |
| Middleware; Permissions Manager | Deploy applications to remote server | 8 | As a manager, I want a solution that is hosted on a remote server so that employees from across the world can utilize its services.<br><br>As an administrator, I want to be able to access the GUI without having to install an application on my computer. |
| Middleware | Logging for Middleware | 3 | As an engineer, I want a solution that is easy to troubleshoot and modify if needed. |
| Permissions Manager | Add tooltips to PM | 2 | As an administrator, I want a user interface that is easy to navigate so that I can complete my job quickly. |
| Permissions Manager | Replace search bars with filters for overview screens | 5 | As an administrator, I want to edit multiple user permissions at once so that I can save time<br><br>As an administrator, I want to limit manual input so that I make fewer mistakes. |
| Middleware | Update Python client | 1 | As an employee, I want to access company databases quickly so that I have the information I need to make informed decisions. |

*Table 7.5 - Sprint 5 Tasks*

We were able to successfully deploy our applications to the remote servers during this sprint. Our team also met with our supervisor in the middle of the week to review the documentation we had so far and received additional feedback. We learned that implementing logging would be useful for everyone if we needed to debug our code. In the meantime, our team also started to configure NTLM authentication while we followed up with the SSO team regarding the status of Kerberos authentication.

During our retrospective/review meeting, we reflected on how the sprint went. Deploying our applications to the remote server was a great success, but testing the application was on hold because we needed to request specific permissions for the user hosting the remote server. Following up with the SSO team about Kerberos authentication resulted in more questions between our supervisor and SSO team representatives, which ended up pushing this process back. Despite certain obstacles, we felt that we made significant progress since we were at least able to deploy to the remote server. Below is a burndown chart (Figure 7.5) comparing the number of points before we began the first sprint and the number of points remaining after each sprint so far.

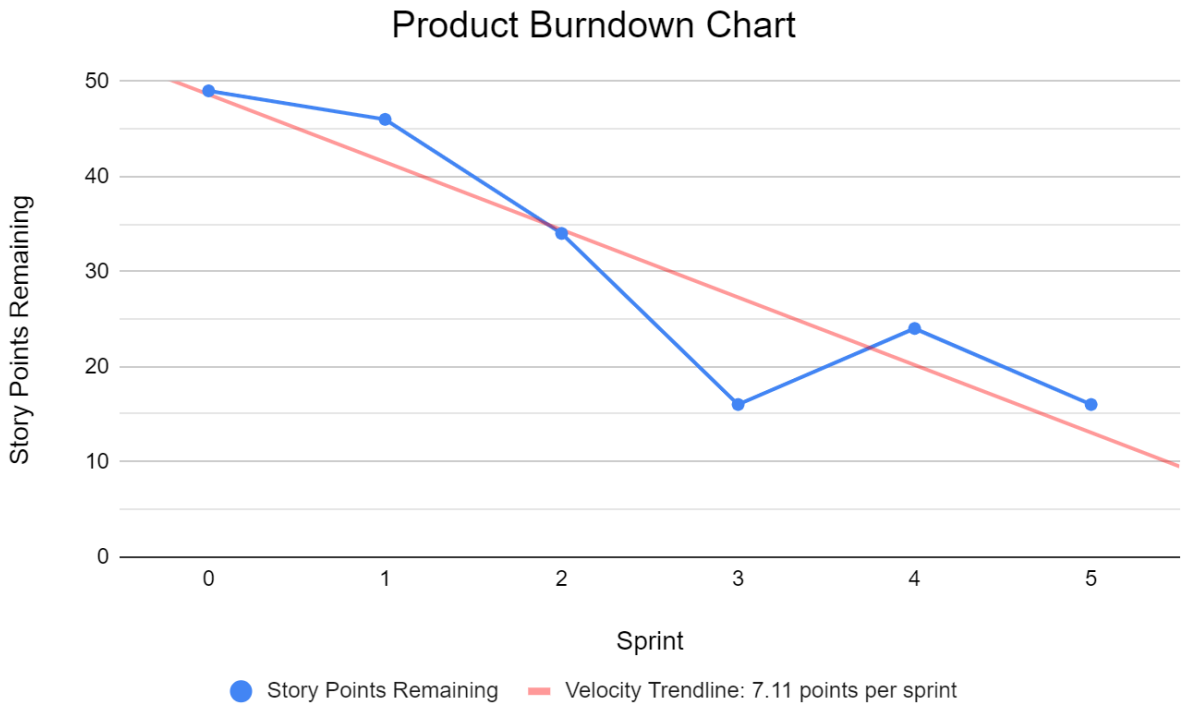


*Figure 7.5 - Sprint 5 Burndown Chart*

## 7.6 Sprint 6: November 25 - November 29

At our sixth sprint planning meeting, we aimed to implement logging and Oracle compatibility for our applications. We also aimed to test our applications on the remote servers to see if we could access another employee's databases. Since this was the week of Thanksgiving break, we wanted to prioritize tasks that we could do that did not involve following up with other employees who would be traveling for the holiday.

Table 7.6 below shows the Trello tasks and the user stories that the tasks pertained to for the sixth sprint. Red rows are Trello tasks that were not previously in our Backlog and were added in during the sprint after our initial demonstration. Yellow rows are Trello tasks that rolled over from previous sprints if they still needed work.

| | | | |
|---|---|---|---|
| Middleware | Authentication System | 13 | As a manager, I want to restrict my employees' access to databases so that they cannot view or manipulate data without explicit authorization. |
| Permissions Manager | Logging for Permissions Manager | 3 | As an engineer, I want a solution that is easy to troubleshoot and modify if needed. |
| Middleware; Permissions Manager | Oracle compatibility | 5 | As an employee, I want to access company databases quickly so that I have the information I need to make informed decisions.<br><br>As an engineer, I want a solution that is compatible with applications written in many programming languages so that I do not have to create a new service for each type of application. |
| Permissions Manager | Refactor management panel components into editor and overview components | 2 | As an engineer, I want a solution that is easy to troubleshoot and modify if needed. |
| Permissions Manager | Replace search bars with filters for editor screens | 5 | As an administrator, I want to limit manual input so that I make fewer mistakes. |
| Middleware; Permissions Manager | Testing to access other employees' databases | 5 | As an employee, I want to access company databases quickly so that I have the information I need to make informed decisions. |

*Table 7.6 - Sprint 6 Tasks*

Due to the Thanksgiving break, we did not hold a formal retrospective/review meeting for this sprint. We were able to log errors from our applications on the remote server, and access another employee's database from the remote server, after additional configuration. We ran into some obstacles implementing Oracle compatibility and trying to fully test our application. We learned that there were no Oracle databases that we could test with that were based in the United States, and that we would have to test with Oracle databases stationed in other countries. This would require us to obtain more information, so we would have to follow up with our supervisor and other employees after the Thanksgiving break.

Although we were able to access another employee's database, certain features of our application did not properly work in the remote server due to similar errors related to the dependency error we kept running into. We tried to investigate how to overcome this obstacle as well as sought out alternatives. With the short sprint time, we made significant progress since we were able to send basic POST requests to the middleware for multiple databases and the permissions manager was able to retrieve database and stored procedure information from those databases. Below is a burndown chart (Figure 7.6) comparing the number of points before we began the first sprint and the number of points remaining after each sprint so far.

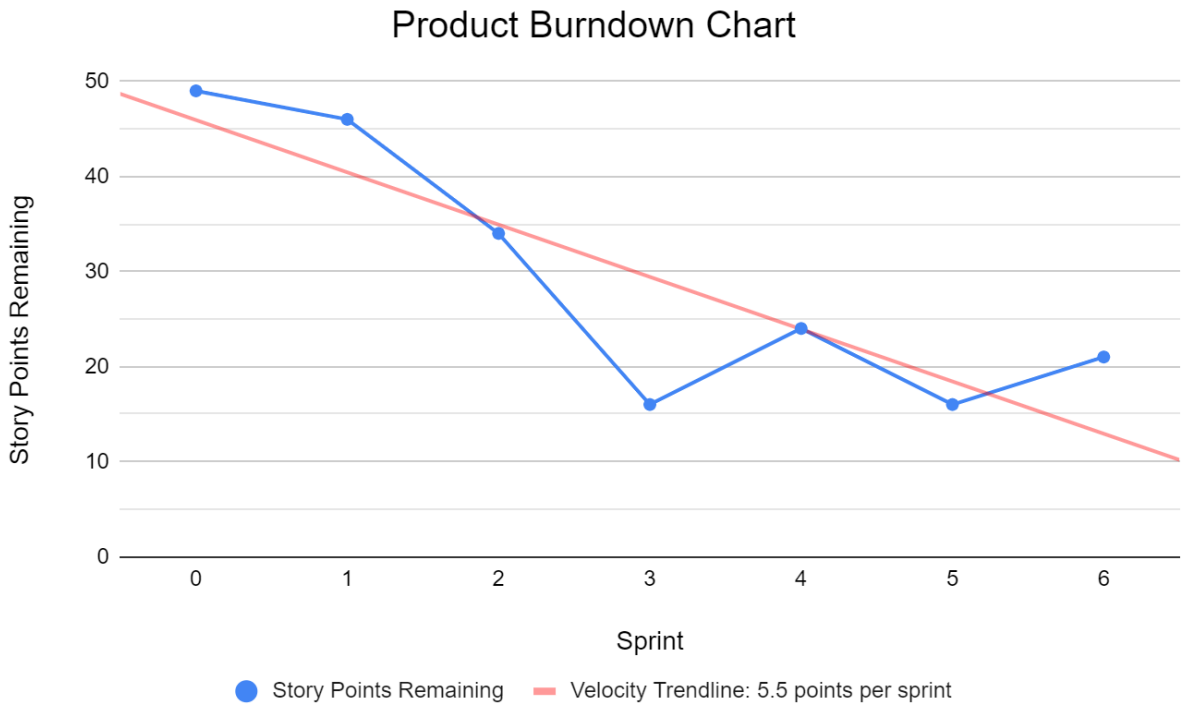


*Figure 7.6 - Sprint 6 Burndown Chart*

## 7.7 Sprint 7: December 2 - December 6

At our seventh sprint planning meeting, we mainly aimed to resolve the dependency error obstacle, since our application would not work properly if we could not resolve it. We also aimed to finish configuring the authentication system since we were almost done following up with the SSO team. Our team had a practice presentation scheduled with our supervisor, as well as code review for the middleware service, so we also aimed to compile PowerPoint Presentation slides and prepared for the code review.

Table 7.7 below shows the Trello tasks and the user stories that the tasks pertained to for the seventh sprint. Red rows are Trello tasks that were not previously in our Backlog and were added in during the sprint after our initial demonstration. Yellow rows are Trello tasks that rolled over from previous sprints if they still needed work.

| | | | |
|---|---|---|---|
| Middleware | Authentication System | 13 | As a manager, I want to restrict my employees' access to databases so that they cannot view or manipulate data without explicit authorization. |
| Middleware; Permissions Manager | Oracle compatibility | 5 | As an employee, I want to access company databases quickly so that I have the information I need to make informed decisions.<br><br>As an engineer, I want a solution that is compatible with applications written in many programming languages so that I do not have to create a new service for each type of application. |
| Middleware; Permissions Manager | Refactor Configuration Manager code | 3 | As an engineer, I want a solution that is easy to troubleshoot and modify if needed. |

*Table 7.7- Sprint 7 Tasks*

Our team was able to overcome the obstacles related to the dependency by migrating to use an alternative service that the company had. We also successfully implemented NTLM authentication for the client applications and middleware service. At our practice presentation

with our supervisor, we received positive feedback and suggestions for what else we could include in the presentation. We also received positive and constructive feedback regarding the middleware service during the code review, and we aimed to address those final changes before the end of the project.

During our retrospective/review meeting for this sprint, we reflected on how we persevered through our obstacles and the amount of progress we had achieved before the end of the project. We discussed presentation strategies and any last changes we wanted to address. Below is a burndown chart (Figure 7.7) comparing the number of points before we began the first sprint and the number of points remaining after each sprint so far.

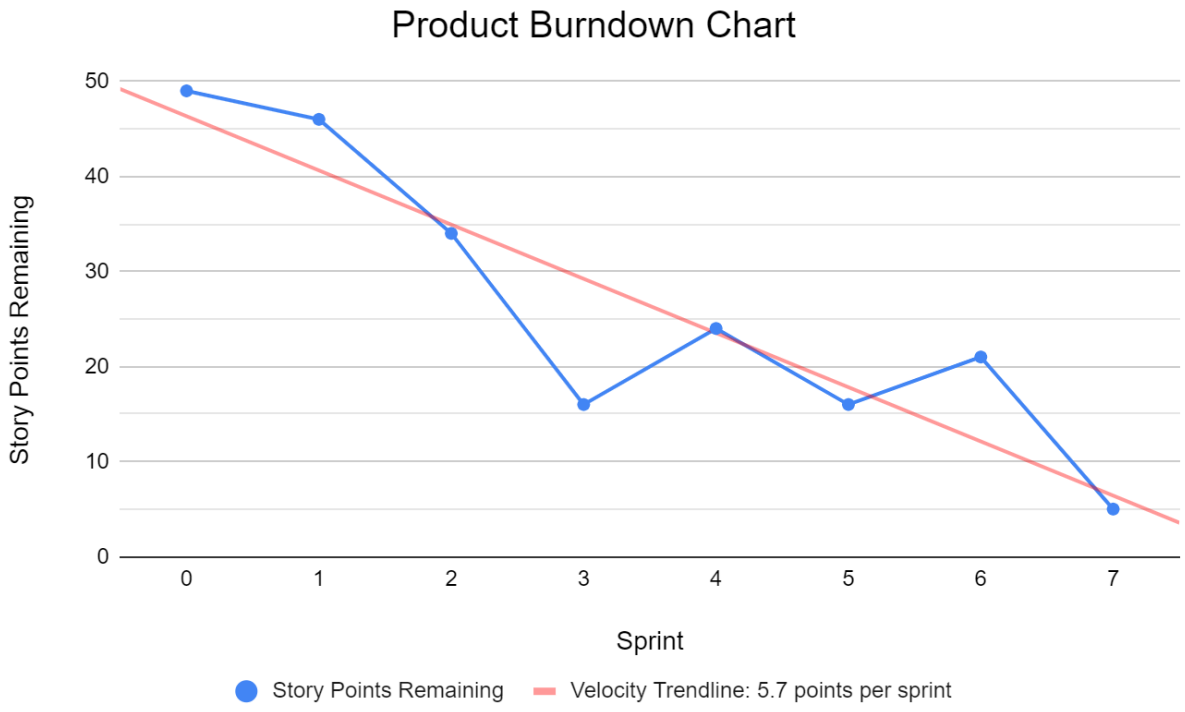


*Figure 7.7 - Sprint 7 Burndown Chart*

## 7.8 Sprint 8: December 9 - December 13

At our eighth and final sprint planning meeting, we planned to finish the remaining tasks before our last day at BNP Paribas. We also had a code review for the permissions manager scheduled at the beginning of this week, and our final presentation was scheduled to be in the middle of the week. Any last-minute feedback from the code review would either be addressed before our last day or documented in the Future Development chapter.

Table 7.8 below shows the Trello tasks and the user stories that the tasks pertained to for the eighth sprint. Yellow rows are Trello tasks that rolled over from previous sprints if they still needed work.

| | | | |
|---|---|---|---|
| Middleware; Permissions Manager | Oracle compatibility | 5 | As an employee, I want to access company databases quickly so that I have the information I need to make informed decisions.<br><br>As an engineer, I want a solution that is compatible with applications written in many programming languages so that I do not have to create a new service for each type of application. |

*Table 7.8 - Sprint 8 Tasks*

We received positive and constructive feedback for the permissions manager and addressed any last changes. We also completed all documentation related to our project and practiced for our presentation. During the final presentation, we presented our project to our advisors, our supervisor, and other BNP Paribas employees by giving a PowerPoint presentation and then a demonstration of what we had completed for the project. The presentation and solution demonstration went extremely well, and BNP Paribas was satisfied with the end result.

At our last retrospective/review meeting, we reflected on how the whole 7-week project went in addition to the last week. We believe throughout this project experience we made recurrently made significant progress and successfully overcame all the obstacles that we encountered, to the best of our abilities. Below is a burndown chart (Figure 7.8) comparing the number of points before we began the first sprint, the number of points remaining after each sprint, and the number of points at the end of the term.

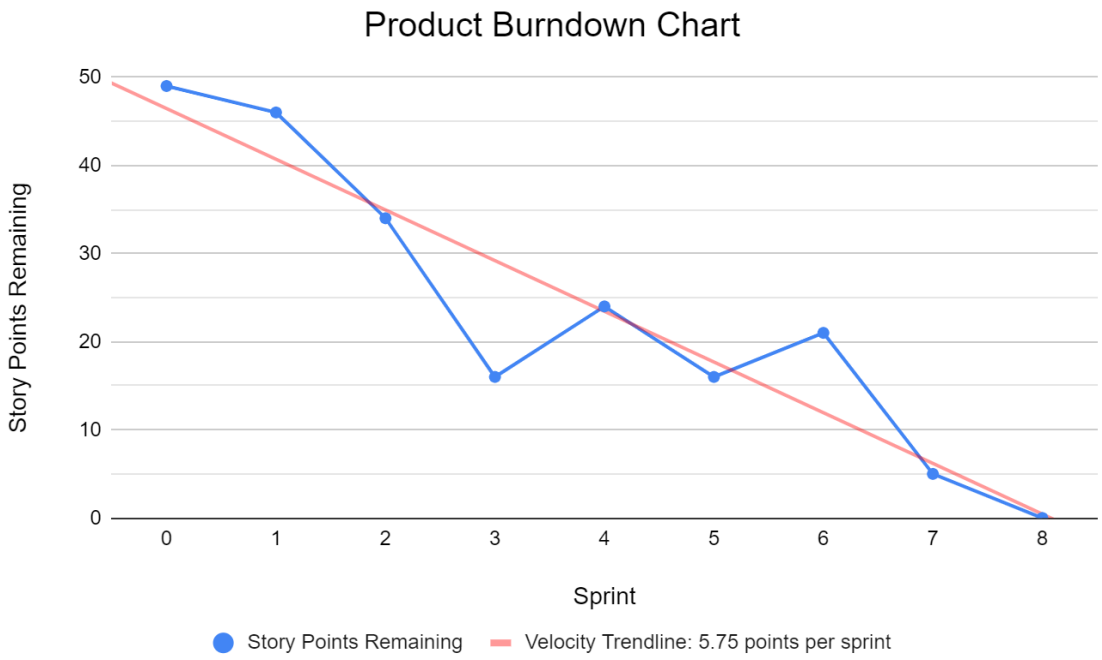


*Figure 7.8 - Sprint 8 Burndown Chart*

# Chapter 8: Assessment

The initial goal of our project was to develop an API between client applications and databases that these applications need to access. This serves to address the current security risks of directly accessing company databases. Our team fulfilled the initial goal by creating a middleware API to authenticate and authorize client applications before accessing data in company databases. We also successfully implemented an accompanying permissions manager to manage what users are authorized to access. Our team realized several achievements and took away many valuable lessons from our successes and setbacks throughout this project experience.

## 8.1 Achievements

In addition to fulfilling our initial project goals, we continued to implement more features for the middleware and permissions manager after creating solutions to satisfy the minimum requirements. We received positive feedback when we first presented our solution to our supervisor and other employees from the bank. They proceeded to provided suggestions for additional features that were not outlined in the initial project goals. These suggestions ranged from an extra add-in for the middleware service and client applications, as well as automatic cleanup of Configuration Manager data. We were also able to successfully implement these components during our time at BNP Paribas to provide better project deliverables.

The journey to complete our project was by no means a straightforward process with no failures. We ran into many obstacles along the way, ranging from configuration issues and errors deploying to the remote server. While persevering through these challenges, we learned a lot about our individual abilities and the abilities of our team as a whole.

## 8.2 Major Takeaways

The 7-week project time frame allowed us to intensely focus on what needed to be completed for our project. During our time at BNP Paribas, we also learned many lessons while working on our project that facilitate our future success. To provide insight to students about to begin a similarly rigorous project or to those about to enter the workforce, our team took note of major takeaways from our project experience.

BNP Paribas had security measures to ensure that the correct users have access to certain applications. Other large companies may also have similar security measures. New employees need to submit an access request for approval in order to be granted and provisioned the applications that they need. Waiting to access applications blocks one from actively working on tasks that relate to that application. In these situations, it is best to try and plan ahead to make requests for permissions or services before they are needed.

We also had the opportunity to work with multiple teams within BNP Paribas. Representatives from different teams provided a variety of knowledge and insight. We also learned that we needed to complete additional processes that involved submitting requests to specific teams so that our project could move forward to the next step even after our project ends. We learned that it is essential when interacting with other teams to explain what is needed as clearly and thoroughly as possible to avoid miscommunication, and to frequently follow up to resolve the process efficiently. If anything is unclear, it is important to clarify all questions to make sure the process is completed correctly the first time.

One final lesson that we learned is that you should take some time to question the methods proposed by others. This takeaway is significant because we should always strive to develop and improve solutions to be as efficient as possible. As newer technologies continue to arise, it is important to examine the way solutions are currently developed and to decide if migrating to newer solutions is a better solution long-term.

# Chapter 9: Future Development

Our solution successfully mitigated the security vulnerabilities posed by direct database access, but given the condensed 7-week timeframe for this project, not all the components of an optimal solution could be implemented. To provide guidance to the BNP Paribas employees and students that will further our development, our team has formulated a set of improvements that should be considered in the future.

## 9.1 Future Implementations for the Middleware Service

Currently, the middleware service can handle requests from client applications that need to access a Microsoft SQL Server database or an Oracle database. Our team has tested the middleware with more Microsoft SQL Server databases than Oracle because there were no Oracle databases based in the United States. In the future, the middleware should be able to support databases other than Microsoft SQL Server and Oracle. If additional cases are needed, a future employee or student can easily create a new class to support the new type of database and implement the generic database interfaces we have provided.

To handle requests for Oracle databases, the client needs to provide the server credentials with the rest of the request body. If credentials are changed in the future, then every client application programmed to request access to the same Oracle server would need to be updated. This would also be the case for other types of databases that the middleware could support in the future. We recommend that developers create a predefined list of servers for the middleware service that holds the related credentials to reduce client maintenance cost. Instead of having to transmit server credentials from client applications, adjustments would only have to be made to the middleware.

## 9.2 Future Implementations for the Permissions Manager

When a client requests user, role, or procedure data, the permissions manager currently fetches the data from its original source. This unnecessarily burdens database servers and increases wait time for users. Employing an in-memory caching solution, such as Memcached, would enable the server to immediately respond to the client with the requested data. Caching introduces the problem of data inconsistency, but user and stored procedure data is not expected

to change very frequently. The role data is probably the most dynamic data set; however, it should exclusively be modified by an instance of the permissions manager. This means that when a user makes a request to update role data in the Configuration Manager, those same changes can be pushed to the cache.

Although our team sought to construct a user-friendly application for managing user permissions, the design was largely influenced by the technical specifications outlined by BNP Paribas' security standards. The application's functionality and user interface were under constant review by several company managers, but it was unclear as to who would be responsible for managing user permissions once our solution is fully adopted. As a result, it was not possible to conduct an accurate study of user's experiences while using the permissions manager. To further enhance the usability of the service, it should be thoroughly critiqued by the system administrators that would be using it. This investigation would provide developers the insight into user preferences for things like colors, layouts, font size, and navigation. Tailoring the application to real users would ultimately deliver a much more enjoyable and productive experience.

## 9.3 Migrating to Kerberos Authentication

The original plan BNP Paribas had for our solution was to implement Kerberos authentication. We found that there needs to be a third-party Key Distribution Center (KDC) configured for Kerberos authentication. We set up a meeting with BNP Paribas' SSO team and our supervisor to learn about how to set up the KDC. Since our solution would be hosted in BNP Paribas' new production environment, we learned that all necessary configuration needed before implementing Kerberos authentication would take a long time and would most likely be finished after our team was expected to complete our solution. Our team continued working with the SSO team to start the Kerberos configuration process, but for our solution we had to utilize a different authentication protocol.

As an alternative solution, our team decided to implement NTLM authentication method. Our team learned that NTLM is more secure than basic authentication protocols because the user does not have to manually enter their username and password. There are other BNP teams that utilized the NTLM authentication protocol, so we decided to configure our solution with NTLM.

Although our solution works with NTLM authentication, we encourage future employees or students to migrate our solution to use Kerberos authentication instead. The NTLM protocol does have susceptibility risks for Man-in-the-Middle attacks because NTLM does not provide mutual authentication (Zinar, 2018). Unlike NTLM, Kerberos authentication does support mutual authentication, which involves the client authenticating to the server and the server authenticating to the client (De Clercq, 2007). Migrating from NTLM to Kerberos will provide more security for the company and strengthen our solution to further eliminate existing security risks.

## 9.4 Utilizing Microsoft SQL Server

BNP Paribas uses an application called the Configuration Manager to store tables related to authorization information. Our middleware and permissions manager needed to use the Configuration Manager to provision user roles and to check if users have been authorized to execute given stored procedures. The Configuration Manager essentially looks like an Excel Workbook where employees can make tables that resemble Excel Worksheets to store related information. To read and write to the Configuration Manager, we needed to use the company's NuGet package. However, this caused our team to run into numerous obstacles during the development process. Since we could not resolve errors on the remote server, we had to migrate from the NuGet package to another service that could access the Configuration Manager.

We discovered from our attempts to use both the NuGet package and the service, however, that querying information was not as efficient as we would have liked it to be. There was no way to reference data by column names, which resulted in iterating through table rows. Our team highly recommends that developers migrate from storing authorization information in the Configuration Manager application to a Microsoft SQL Server database instead. This transition would require developers to modify the middleware and permissions manager quite a bit, but using a Microsoft SQL Server provides a lot of advantages over the Configuration Manager. With Microsoft SQL Server, an engineer could utilize query optimizers to expedite searching, and they could include constraints to make the system more robust and prevent users from intentionally or accidentally clearing cells of information. There would also be fewer obstacles using a Microsoft SQL Server because it does not require outdated NuGet packages or rely on internal services.

## 9.5 Testing

The solution that ultimately brings BNP Paribas' client application in compliance with its IT security policy will need to be capable of handling requests from thousands of users simultaneously. Multiple users within one team can send requests to our middleware service that is hosted on the remote server, but we recommend that developers test it more thoroughly with large user groups.

These tests could be furthered by examining how our solution behaves under both normal and peak conditions. It is critical that our solution does not become overloaded with requests. Depending on the number of users, it may be wise to deploy multiple instances of the middleware and permissions manager to multiple servers. These servers may run their own instances of our solution with a load balancer to distribute network traffic and increase capacity.

If our solution is augmented in the future, we encourage implementing regression testing to ensure that additional features do not break the current working solution. When our solution migrates to a worldwide audience, it is essential that improvements to our solution do not cause existing features to stop working. Overall, testing is a very important component of application development and one must be taken seriously when it comes to large scale applications.

# Chapter 10: Conclusion

A multitude of BNP Paribas' internal services rely on data that resides in Microsoft SQL Server and Oracle databases. Most of these applications were developed following a two-tier architecture in which the client directly interfaces company databases. Once BNP became aware of the security vulnerabilities that emanate from this approach to data access, the company adopted an IT security policy that forbids direct database access from user facing applications. The goal of this project was to help BNP Paribas comply with its IT application security policy by developing a universalizable middleware that serves as a mediator between breeching applications and databases that they need to access.

In fulfillment of this requirement, our team constructed a centralized authentication middleware that enables applications to be decoupled from databases. Instead of requesting data directly from the databases, this new access model simply requires client applications to access data via a middleware service. The main functionality of the middleware includes authenticating users with NT LAN Manager security protocols, authorizing users with data from BNP's Configuration Manager, and executing stored procedures. This highly versatile service is compatible with any application capable of sending HTTP requests and processing JSON. With regards to database compatibility, the middleware is currently capable of executing stored procedures that reside within Microsoft SQL Server and Oracle databases. Nevertheless, the service was developed following the factory design pattern, which would enable developers to easily onboard additional DBMS platforms.

In addition to the middleware service, our team also designed and developed a web application that system administrators can use to efficiently manage which stored procedures a user has permission to execute. While it may take BNP Paribas several years to become compliant with its IT security policy, we hope and expect that our middleware and permissions manager will serve as the foundation for the solution that is implemented company-wide.

# Bibliography


About the Group. (n.d.). Retrieved from https://group.bnpparibas/en/group

ActiveCollab. (2019). Self-Hosted Project Management. Retrieved from https://activecollab.com/self-hosted-project-management

Activities. (n.d.). Retrieved from https://group.bnpparibas/en/group/activities

Agilemanifesto (2019). Retrieved from https://agilemanifesto.org/

ASP.NET. (n.d.). Retrieved from https://dotnet.microsoft.com/apps/aspnet

BNP Paribas. (2016a). History: two centuries of banking. Retrieved from https://group.bnpparibas/en/group/history-centuries-banking

BNP Paribas. (2016b). Our strategy and corporate culture. Retrieved from https://group.bnpparibas/en/group/strategy-corporate-culture

BNP Paribas. (2016c). Start-ups which improve banking and finance. Retrieved from https://group.bnpparibas/en/hottopics/fintech/briefing

BNP Paribas in the U.S. (n.d.). Retrieved from https://usa.bnpparibas/en/bnp-paribas/bnp-paribas-us/

Bruegge, B., & Dutoit, A. H. (2010). *Object-oriented software engineering: using Uml, patterns, and Java*. Boston: Prentice Hall.

Consolidated Financial Statements [PDF File]. (2019). Retrieved from https://invest.bnpparibas.com/sites/default/files/documents/bnpp_ef_31.12.18_eng_0.pdf

De Clercq, Jan. (2007). Comparing Windows Kerberos and NTLM Authentication Protocols. Retrieved from https://www.itprotoday.com/security/comparing-windows-kerberos-and-ntlm-authentication-protocols

Eeles, P. (2006). What Is a Software Architecture? Retrieved from https://www.ibm.com/developerworks/rational/library/feb06/eeles/index.html

Kanbanize (2019), Kanban Explained for Beginners. Retrieved from https://kanbanize.com/kanban-resources/getting-started/what-is-kanban/

N-tier architecture style. (n.d.). Retrieved from https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier.

Olic, Aleksander. (2017). Self-Hosted vs Cloud-Based Project Management Tool. Retrieved from https://activecollab.com/blog/product/self-hosted-vs-cloud

Sarycheva, Yana. (2019). Waterfall Model in SDLC. Retrieved from https://xbsoftware.com/blog/software-development-life-cycle-waterfall-model/

Rajkumar. (2017). Software Architecture: One-Tier, Two-Tier, Three Tier, N Tier. Retrieved from https://www.softwaretestingmaterial.com/software-architecture/.

The Client/Server Model. (n.d.). Retrieved from https://www.ibm.com/support/knowledgecenter/en/SSAL2T_8.1.0/com.ibm.cics.tx.doc/concepts/c_clnt_sevr_model.html

Trevellyan, Suzanne. (2017). Hosted vs Self-Hosted Websites. Retrieved from https://trevellyan.biz/hosted-vs-self-hosted/

What is IIS Express? How It Works, Tutorials, and More. (2017). Retrieved from https://stackify.com/what-is-iis-express/

What is Three-Tier Architecture? - Definition from Techopedia. (n.d.). Retrieved from https://www.techopedia.com/definition/24649/three-tier-architecture.

What is Two-Tier Architecture? (n.d.). Retrieved from https://www.techopedia.com/definition/467/two-tier-architecture.

Visual Paradigm (2019), What is Agile Software Development. Retrieved from https://www.visual-paradigm.com/scrum/what-is-agile-software-development/

Zinar, Yaron. (2018). The Security Risks of NTLM: Proceed with Caution. Retrieved from https://www.preempt.com/blog/the-security-risks-of-ntlm-proceed-with-caution/