Academy of Economic Studies, Bucharest

Information Security Masters

An analysis of software applications in financial institutions

dissertation

Coordinator

Prof. Univ. Dr. Cristian-Eugen CIUREA

Graduate

Cristian-Mihail DUMITRESCU

Bucharest 2024

Declaration regarding the originality of the content and assumption of responsibility

Through my presence, I declare that the presented results in my work are my own results except for the case where I refer to other authors. I confirm the fact that any material use from different sources (magazines, books, and websites) are as clearly as possible referenced in the dissertation and are indicated in the bibliographic references.

Contents

Chapter 1 – Introduction.

As the title suggests, this paper presents the possible security problems that may arise when designing, configuring, and using IT (annex 2) infrastructure, especially software in a financial institution. The study described in Chapter 2 noticed some problems regarding developer behavior regarding the research and knowhow about security topics that can affect their everyday work. The study and this paper are accompanied by a project, which showcases a possible solution to software problems, through developer training.

The conclusion of the study gives meaning and motivation to the created application, as an internal tool.

The study also aims to research the trust of the everyday individual for the security of the systems of banks, and the care of the developers that work on them, from the security standpoint. This study alongside other relevant research papers and books are the contents of Chapter 2.

Chapter 3 is comprised of a description of technologies and programming languages used when creating the problem-solving application.

A concise, but very important definition of architecture, alongside diagrams and schemas of different used systems is described in Chapter 4. Here the general structure of a financial institution's IT (annex 2) systems is also presented.

The last two chapters, Chapter 5 and Chapter 6 comprise a presentation of the application implementation, small code snippets where necessary and a small conclusion in which possible additions to the application or to the study are discussed.

For the application creation, a few notable tools have been used:

- React [1]
- .NetCore [2]
- JSON [3]
- HTML [4]
- CSS [5]
- JavaScript [6]
- Sqlite3 [7]

The first thing that this paper wants to clarify is the definition of "Financial institution": The United Stated Government through the Financial Crimes Enforcement Network [8] defines a

financial institution as any person (as in individual, business, partnership, trust, or estate) doing business in one or more of the following capacities:

1. Bank
2. Broker or dealer in securities
3. Money services business
4. Telegraphy company
5. Casino
6. Card club
7. A person subject to supervision by any state or federal bank supervisory authority

For the context of this paper, I will be referring to financial institution as a business that has the capacity of a bank, a broker/dealer in securities or a money services business.

There are two types of threat actors: insider and outsider threats. While the outsider threat seems like a more dangerous problem, at first glance, the insider threat has received a lot of attention and is cited as a very serious problem in many studies, such as in the "Insider Threats in Cyber Security" [9].

The insider threat is very prevalent because most insiders by their role in the system may have already be in a position where some security systems are already bypassed and in the worst-case scenario, the systems now rendered useless may be relied upon when designing the rest of the security. Take for example this article [10], in which it is said that employees of Tesla entrusted with certain data, misappropriated the information to a German news outlet.

The insider problem is solved with proper background checks for all parties involved in the IT (annex 2) system and with proper training, these types of security problems can be reduced to a minimum.

The outsider threat is solved with well implemented security systems and protocols, by implementing proper training within the organization, some outsider threats such as "phishing" and "malware" are neutralized.

There are four main systems that interact with the client of a financial institution. There is proprietary hardware facing the client, such as ATMs (annex 2) and POSs (annex 2), there is proprietary software such as websites of said financial institution and there are third party software and third party hardware facing the client such as network equipment in a subsidiary.

ATMs (annex 2) are physically monitored and guarded, while POSs (annex 2) are monitored remotely and can be deactivated if the behavior is uncharacteristic. These devices are audited and controlled periodically to ensure the safety of the companies using them and of the public.

In subsidiaries there are also network devices that must be secured both physically and logically through periodic audits or inspections and through the creation of logical boundaries with the use of firewalls and Virtual LANs (annex 2).

For the purposes of web sites or web applications facing the clients, the security requirements are higher than ever. The physical mode of transport as well as the network must be secured, in addition to the security requirements of the banking API (annex 2) system.

Although IBM X-Force Threat Intelligence Index 2024 [11] at page 9 say the phishing attacks have come down, it reflects the continued adoption of phishing mitigation techniques. In 2016, the same research said that almost 60% of the attacks come from within a company.

To mitigate such problems, training and accountability for all employees is mandatory. When dealing with financial institutions, if too many employees do not respect guidelines and procedures, an attacker may use the small gaps in security for a greater attack which can lead to catastrophic circumstances.

Data safety is also important. Employees must keep their own data and accounts private, as well as the data and account details of their clients.

Chapter 2 – Literature review of security in financial institutions and case study.

The subject of security in financial institutions, especially in banks, is not a new one. There has been extensive research done to analyze and help prevent robberies and theft on any level. Some of the crime has therefore moved from the institution towards the end user, as described in the conclusion of "Banking information resource cybersecurity system modeling." [12].

The security of banking apps has therefore become a great priority, as presented by "Cybersecurity in Banking and Financial Sector: Security Analysis of a Mobile Banking Application" [13]. This paper does not focus on the user side of the equation when it comes to apps, because it is near impossible to keep the app safe, users having the ability to do as they please with their devices and data. It will discuss the API (annex 2) security challenges faced by the institution.

There are enough factors that top management in banks, as well as the public must understand and what should be assessed such as the ones in the conclusion of "Assessing the Factors of Cybersecurity, Awareness in the Banking Sector" [14]. As the recommendations of the paper suggest, proper support and budgeting of the cybersecurity teams is a must.

The physical security and network security are also important as stated by This study about "Bank Robberies and Physical Security in Switzerland" [15] and "Security Issues on Banking Systems" [16].

Chapter 2.1 – Physical security.

Before all the information systems and all the training required to operate them safely, the only issue that existed was physical security. In the current day and age, the physical security of a system may be overlooked when analyzing it.

The preventative measure of physical security cannot be overstated. This study about "Bank Robberies and Physical Security in Switzerland" [15] on the effectiveness of physical security on bank robberies in Switzerland, in chapter 2 and 3 emphasizes the preventive effect of physical security. The study shows that the number of attacks does not necessarily reduce, but the effectiveness does. This idea can also be applied to network and software security as well.

There are many design choices that go into the construction of a physically secure place. The book "Effective Physical Security" [17] presents some of those concepts that are important

even in the context of IT (annex 2) security. Of all the design issues discussed there, for the context give, there are some of underlining importance:

In the chapter 3 about physical design, there is a quote about the level traffic influencing the level of crime negatively, quote of Schlomo Angel, in 1968 [18]. The author adds that the physical design may allow crime to arise through the lack of potential witnesses while providing enough victims.

This knowledge can be used when designing the physical circumstances of an information system. These circumstances should be designed with the four-eyes principle in mind.

Chapter 7, under Use of locks and in chapter 11 Access control systems and Identification badges, describe ways in which space can be separated with the use of authentication/identification. A lack of authentication at the physical level can lead to unauthorized access to information systems which can lead to a security breach.

All the issues presented above become critical when neglected. They may impact the security of any IT system if a vulnerability is found and exploited, or if the infrastructure associated becomes vulnerable to a physical attack, at network or application levels.

Chapter 2.2 – Network security.

Physical security being established, the information transportation security can start being discussed. The chapter not only discusses transport layers, and transport equipment, but also about safety on the user machine.

This book about "Network Infrastructure Security" [19] by Angus Wong describes a lot of the attacks that may appear on a nonsecure network.

According to a study on "Evaluation of Computer and Network Security Strategies: A Case study of Nigerian Banks" [20] , in chapter 3.5, 37% of correspondents noticed some attacks and 15% experienced often attacks on their respective bank's equipment or network. In the same publication, there are some ways to prevent and avoid attacks. The use of well implemented password management, antiviruses, firewall solutions and the use of intrusion detection or intrusion prevention systems are great ways of stopping most network attacks.

This [21] security report from the NSA-National Security Agency outlines a lot of network changes that are required to operate a network secured. Besides the changes outlined above, the

report dabbles into virtual private networks, virtual local area networks, ports, proxies, and many other necessary points of interest for the security of the network.

Chapter 2.3 – Software security.

With the mode of transport established and secured, the discussion can now move towards the software applications that run on the established infrastructure. The applications can either be made in-house or be outsourced, but all of them must have a certain level of security so no negative impact is dealt to the institution.

The cost of cybersecurity breaches in the form of software exploits or "hacks" are high, as presented in "The economic cost of cybersecurity breaches: A broad-based analysis" [22] conclusion. The paper also suggests that the software problems may affect more than the attacked entity.

Database security is at the limit between system administration and application development. The database must be secured, and the applications must be able to modify data unimpeded. This "Database Security—Concepts" [23] describes, in chapter 5, about the challenges of databases in the context of cloud.

Beside those, there are challenges about the way accounts and groups are set up in the way for apps to work properly and securely.

No SQL databases may be of help on the development side, but they are also vulnerable to such attacks, but also some of their own issues, as highlighted in the 4'th chapter of "Security Issues in NoSQL Databases" [24].

For the rest of the paper, an API (annex 2) is the interface between two applications, except for databases. The applications in question can be either monolith or microservice architecture in nature [25].

Over time, APIs evolved and gathered more and more importance, as described in the first 4 chapter of "Cyber Security in API Economy – Issues and Challenges" [26]. As such, the importance of their security increased.

There are many attacks that can happen at an API's level. The "API Security in Action" [27] by N. Madden presents a big catalogue of attacks that are suitable for general purpose API's, such as injection attacks, cross site scripting attacks and ways to prevent them through proper input validation.

The security issues of the API's are problematic in the internal applications of the bank, but also in the external facing facilities such as ATMs (annex 2) and banking apps "Understanding Security APIs" [28] describes in chapter 2 the development of security in such facilities and some attacks in chapter 3.

Chapter 2.4 – Case study.

Based on the literature review of this paper, the consensus is that all systems of a bank must be monitored and protected to ensure data integrity and a functional business model.

In the case of physical and network security, although not easy, the way of conducting checks on it is more transparent than in the case of software. Software often comes from a third party which restricts access to its codebase, for business-related reasons. The surveys described below aim to test the trust or distrust of the every-day person about banking software systems and for the care of developers, in the financial sector or not, for their work, from the security perspective. For such a task, there were designed a couple of surveys destined to either of the groups (every-day people and developers) to see their opinions on the subject. The surveys may be treated separately, with the results coupled together to paint a greater picture of the perceived security of the software systems.

1. The public, every-day person survey has the following question:

- Have you ever considered data protection in your life?
- Do you feel like you are a person prone to having your personal data stolen?
- Have you taken any steps to protect your data?
- Do you feel like the bank you use is protecting your data properly?
- Do you feel like your bank has a good interest in protecting your data?

2. The developer survey has the following questions and possible answers:

- Do you research security issues for the languages and apps you currently use?
- Are you involved in discussions about the level of security of the apps you develop?
- Are you interested in the security of the applications you create?
- Are you interested in the security of data for the applications you develop?

All with the following possible answers: Yes, No and Somewhat

At the end, both surveys have a section where the person has to rete their own familiarity in some security subjects:

- SQL injections
- CORS attacks
- XSS attacks
- Adware
- Spyware
- Session hijacking
- DOS/DDOS attacks
- Buffer overflow attacks
- Code injection attacks
- Impersonation

With the following possible answers:

- Unfamiliar
- Somewhat unfamiliar
- Neutral
- Somewhat familiar
- Familiar.

The population for the public questionnaire was a mixture of people from urban cities of Romania. The people surveyed had a lot of contact with banking systems and had used the services.

The population for the developer questionnaire consisted of people working as developers at different companies in Bucharest and that have finished a bachelor's in the IT (annex 2) field or a master.

The public questionnaire highlights the uninformed public's opinion of the banking system for data privacy and safety, while the developer questionnaire describes the interest or disinterest in the subject of software security of the people that make it.

The restrictions and limitations of the research are given by the sample location of the survey. The problems outlined in the "Results and discussion" section may be just local problems to the surveyed region. Another possible problem may be that the developers surveyed are not necessarily working in the banking sector, so the results may not be totally representative of that sector.

Based on the survey proposed above, the data shows a good interest from the public towards their own data protection and a high trust in the security of the banking system, which was to be expected, since everyday people expect reputable institutions.

A high percentage of the surveyed individuals have considered data protection and have taken steps to protect their own data with moderate success as shown in figure 1.

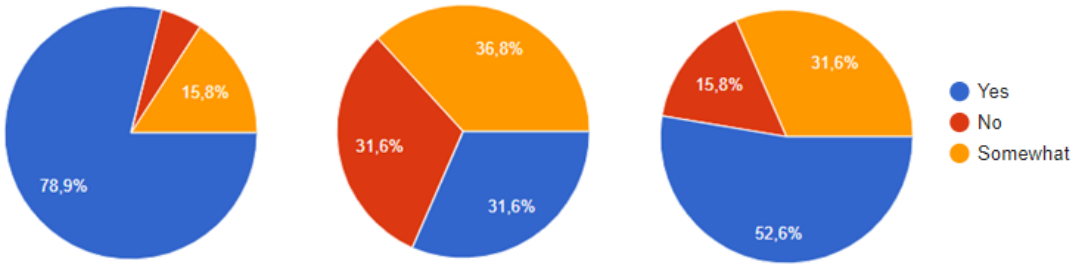

Fig. 1 Response percentages to the first 3 questions of the public survey

Figure 1 represents the data distribution for the answer of the first three questions of the public survey.

The next two questions in the public survey refer to the trust of the public in their own banking provider's ability and desire to provide data protection for them. The answer percentages are displayed in figure 2 and display a high trust in the banking systems.

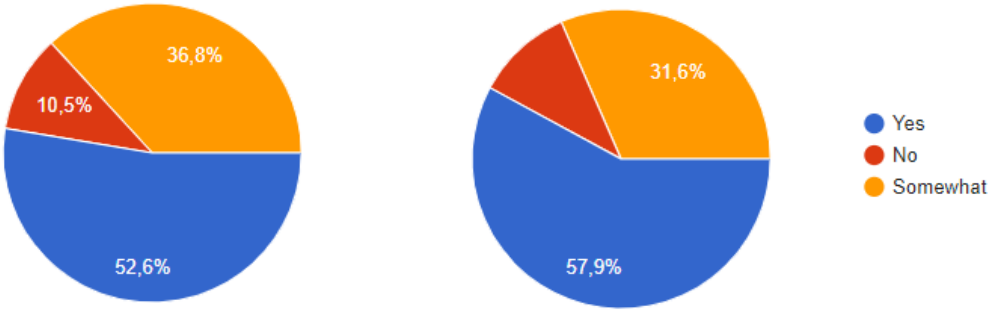

Fig. 2 Response percentages for questions 4 and 5 of the public survey

Another good point of interest of the survey is the familiarity of the everyday person with concepts from the software security sphere of influence.

The data for this part of the survey suggests that a few everyday people were interested in some areas of security as shown in figure 3. Particularly, the area of interest consisted of mostly non-technical and broader subjects such as “Adware”, “Spyware” and “Impersonation”.

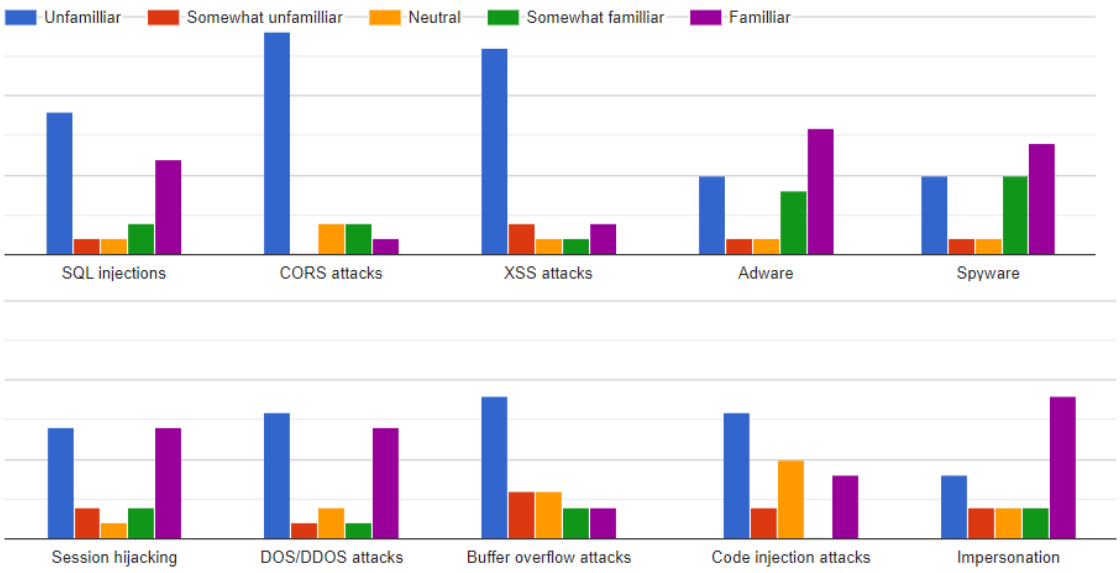

Fig. 3 Response distributions to the familiarity part of the public survey

The survey presents developers as motivated individuals with an interest in the security of the application they are developing and that are personally involved in the security level of the applications they make.

In figure 4, there are the answers to the first two questions of the developer questionnaire.

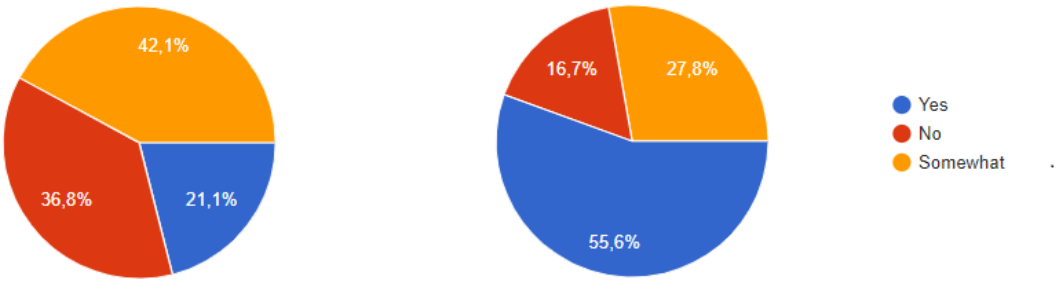

Fig. 4 Response percentages to the first 2 questions of the developer survey

The graphs show that developers are involved in the development of the security features of their applications, but that they mostly lack the drive to do research in this area.

The third and fourth questions, displayed in figure 5, also show that developers are interested in the security of their applications and the safety of their data.

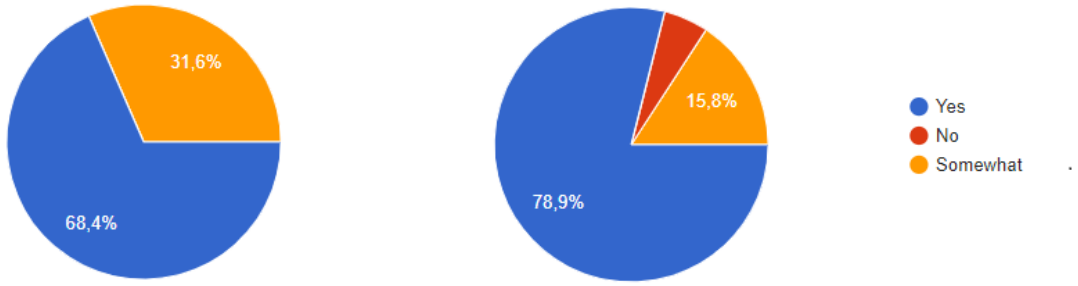

Fig. 5 Response percentages to the third and fourth questions of the developer survey

The interest, but lack of action determined by the first, third and fourth questions, show that developers may need help with resources about the security of the code they implement.

The last part of the survey is shown below, in figure 6. It shows that developers have more familiarity with technical subjects than the everyday person.

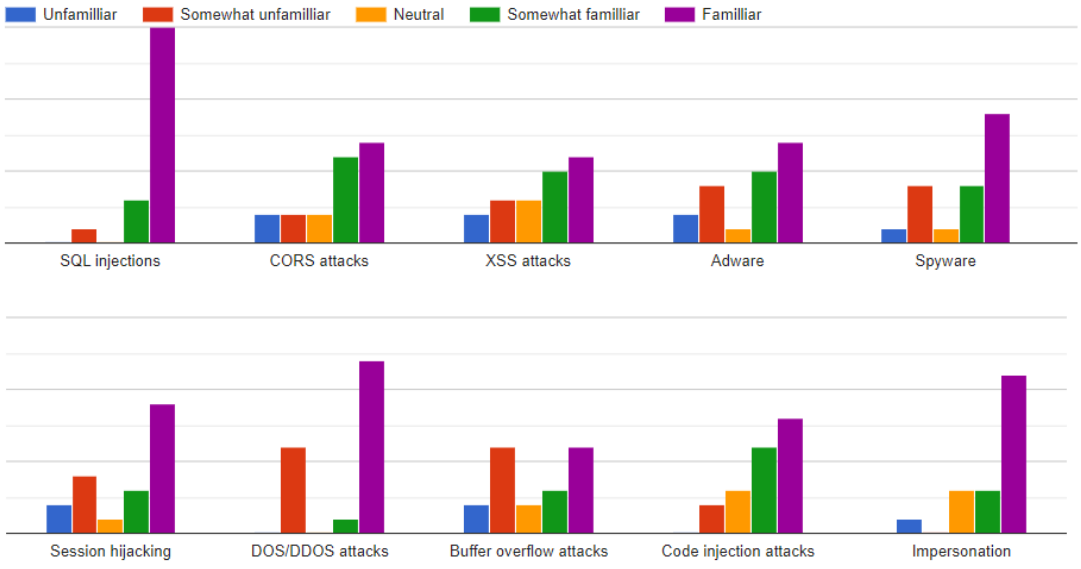

Fig.6 Response distributions to the familiarity part of the developer survey

The familiarity of them with the more mediatized subjects such as “Adware”, “Spyware” and “Impersonation” is alike the general publics.

As stated in the “Customers’ perception of cybersecurity threats toward e-banking adoption and retention: A conceptual study. In ICCWS 2020 15th International Conference on Cyber Warfare and Security (Vol. 270)”’s [29] conclusion, the cybersecurity threats have an impact on the customer’s decision when it comes to banking services. In this regard, the current opinion according to the survey shows good trust in the banking’s security features for software. The study can also be considered a check for “Risk Assessment of Computer Network Security in Banks” [30]’s conclusion with a public opinion survey and about software security, where the author advises banks to enhance their computer system security.

A possible remedy to the problem of developers not researching possible security issues for the systems they design would be a raise in awareness about the possible problems that may arise or the tightening of requirements for development positions.

The raise of awareness can be made by having better documentation of packages that treat security issues, better training or even an AI (annex 2) implementation that detects possible problems and warns developers, such as “Copilot” [31].

For the purposes of this paper, the method chosen to be used does not implement any AI (annex 2) capabilities, since the use of such technologies in the educational field is controversial, as the conclusion of “The Re-examination of the Dangers and Implications of Artificial Intelligence for the Future of Scholarship and Learning” [32] suggests. The method chosen consists of an internal usage application that can help train developers to not create possible security issues through their code.

Chapter 3 – Tools, technologies and programming languages used.

This chapter takes on the task of describing the tools, technologies and programming languages used for the purpose of creating the solution application to the problem described in Chapter 2. The "solution application" from this point forward refers to the combine functionality of the three components: the client application, the server application and the database.

The subchapters below present such tools, separated into the three main components of the application that communicate between themselves.

Chapter 3.1 – Database tools.

The database used for this application is "sqlite3" [7], but any other relational database may be used. Any non-relational databases suffer from the same security issues that relational ones do: the application reading or writing has control to modify or erase data through an account that can be highjacked through any method at application layer. Any account or group permission problems shall be resolved at the network/database administration level, to not allow unauthorized access personnel to modify or delete database entries or, in a worst-case scenario to alter or drop tables, rendering applications using the database useless.

Another problem with databases is the backing-up process. This is another configuration problem for database administrators that must understand the business logic and functionality to know when a database backup may occur to not slow down important or business critical functions. This application does not dabble into administrative work for a Database, and such issues are not of interest or tackled.

A way to edit an "sqlite3" [7] database is a database browser such as "DB Browser for SQLite" [33]. There are many other DB (annex 2) browsers that can be used to edit a database. This database browser has been used because it allowed an easy edit of tables and table entries without the use of the SQL (annex 2) language. The interface for it is very friendly and looks as in figure 7.

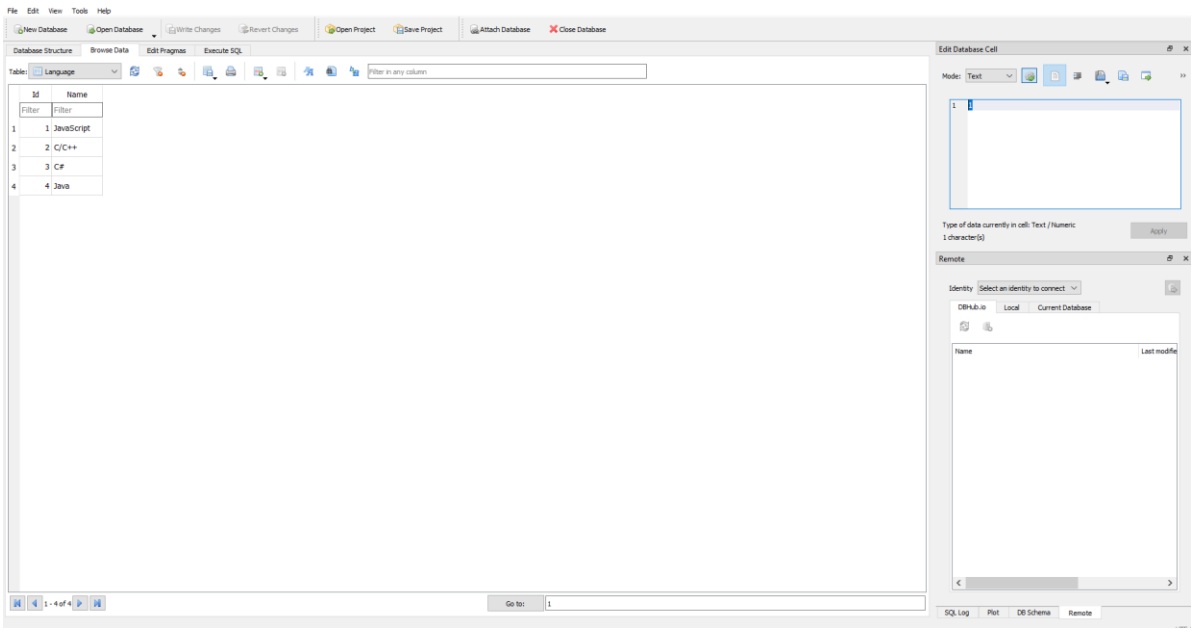

Fig.7 Application used to edit database

Chapter 3.2 – Server application tools.

The server application's tools are explained in this subchapter.

The implementation of the server application has been made in the C# language. It has the form of an API (annex 2) console application that can be run on any Windows [34] systems that has the required package versions. The ".NetCore" [2] is the environment on which the C# program runs and it is the reason why the application can be run on any Windows [34] systems that has the required package versions. It also takes care of the memory usage of the program by automatic allocation and deallocation. This enables the rapid development of applications. Note that the application may be runnable under Linux systems if built correctly and packages are eligible.

The API (annex 2) has the logic for its responses in the controller and the template for the received and sent classes separated in class files.

Some dependencies of the server application include:

- Microsoft.Data.Sqlite
- Newtonsoft.Json
- Swashbucle.AspNetCore

These can be inspected in the Visual Studio environment under Solution Explorer and are depicted such as in figure 8.

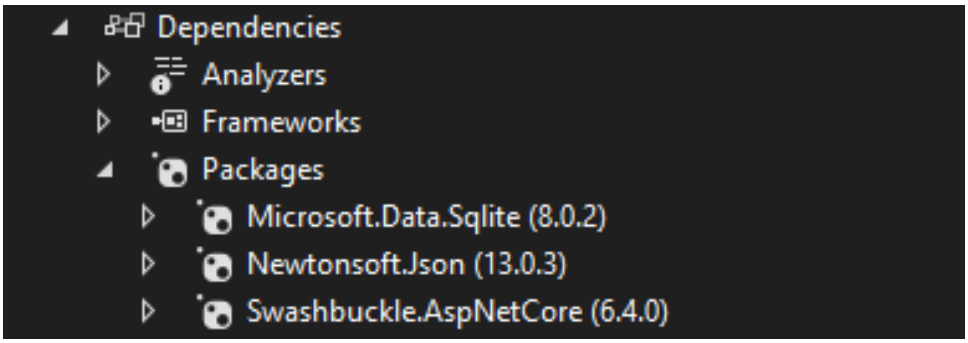

Fig.8 Dependencies for the server application

For the development of the application, the Visual Studio [35] environment has been used. As for the client application, any other text editor along with a console environment may be used to develop the application, but Visual Studio [35] allows the application creator to use a web API (annex 2) project template when creating it to shortcut the installation of packages. The application can be chosen as shown in figure 9.

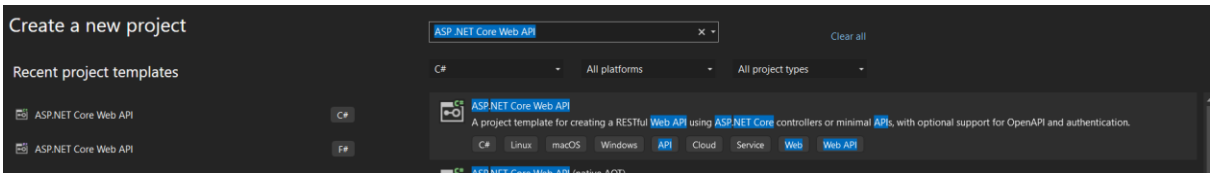

Fig.9 Choosing API application type

Chapter 3.3 – Client application tools.

This subchapter tackles the task of describing the tooling necessary for building the client application.

The client application is created using the “React” [1] library for the easier upkeep of state management. This is a JavaScript [6] library that allows for flexibility by giving the ability to create reuseable components. This ability, alongside already made and available components described below within the Material UI [36] library, gives the ability to rapidly develop a web application.

For the purposes of a front-end client application similar to this, any other library such as "Express" [37] or even vanilla JavaScript, HTML, CSS (annex 2) may be use to ensure that the communication to the API (annex 2) is proper.

The language used for the development is JavaScript, HTML and CSS, although they are not directly used, but rather compiled to by the "React" library. Core concepts of the languages are required, still, because react and the other libraries used only enhance the development uptime by reuseable components that behave in a predictable manner, with the possibility of extension or even creating new ones with raw HTML tags.

Besides the main framework packages, during the development of the client application, several different "support" packages have been used. These "support" packages helped with the implementation of already made components with useful features that aid with the development time.

Packages used include:

- Material UI.
- React tabs.
- React toastify.

These packages can be found in the "package.json" file. A full list of packages with all their required sub-packages are available in the "package-lock.json" file. The general layout of the file can be found in figure 10.

Fig.10 File structure of “package.json”

For the development of the application, the Visual Studio Code [38] environment has been used. Any other text editor along with a console environment may be used to develop a front-end client application.

Chapter 4 – Application architecture, diagrams and schemas.

This chapter presents the application architecture, diagrams and schemas of the three components of the solution application. Besides this subject, the general structure of financial institution's IT (annex 2) systems are also presented.

Architecturally, the application is not too complicated. As we can see in figure 11, the application has a Client-Server-Database structure, where they all communicate. The client application sends HTTP (annex 2) messages to the server application and receives messages alike, while the server application and database communicate through the database driver.

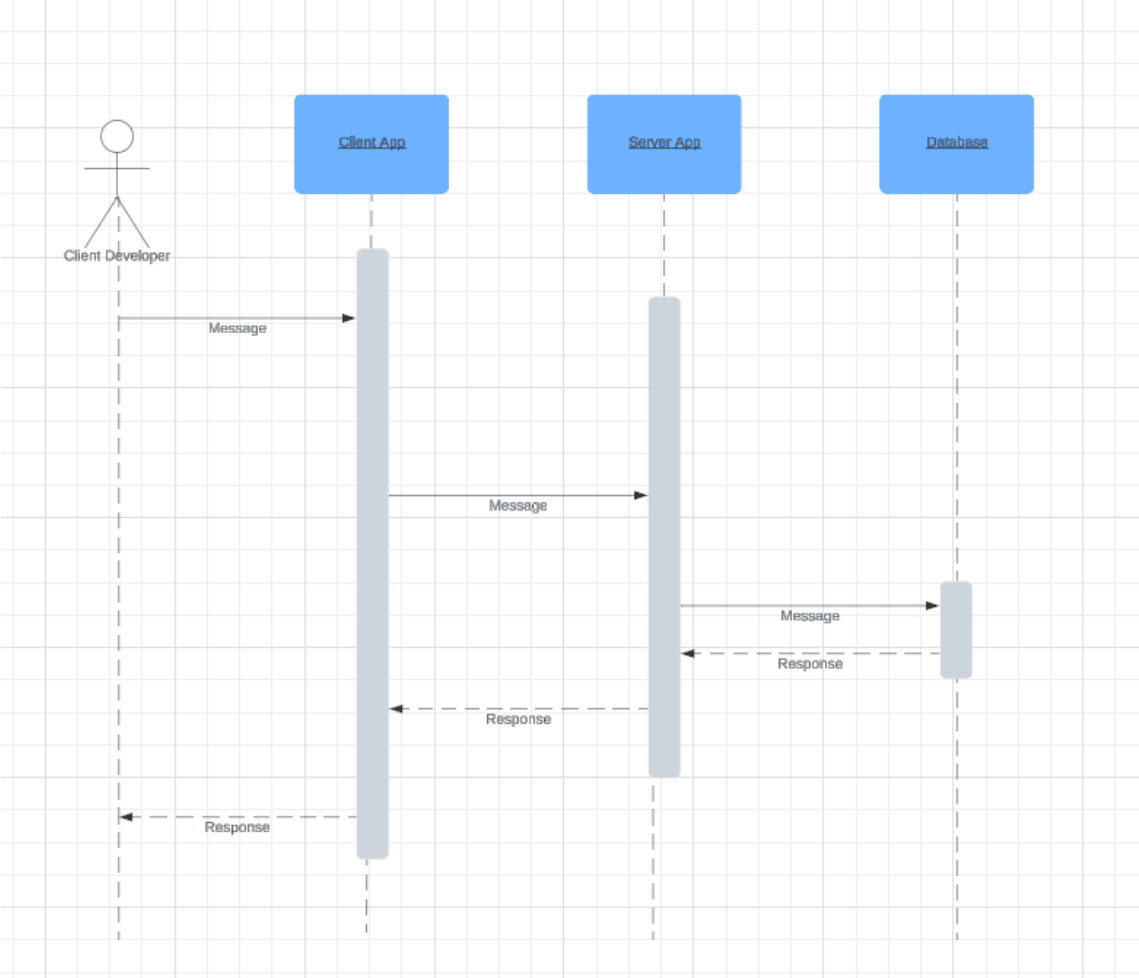

Fig.11 Sequence diagram

Chapter 4.1 – General structure of financial institution's IT systems.

In this subchapter, I will be describing the internal structure of the critical internal systems of a financial institution, mostly Databases and Networks.

Databases are a critical piece of any infrastructure. For financial institutions, reputation plays a big role in their business. Thus, client data must always be kept safe. A good rule, for any organization, is to have backups online and offline, as well as onsite and offsite. Since this paper tackles financial organizations such as banks, data stored is of high value to the institution. Such data shall always be as protected as possible. Keeping backups will lead to an easier recovery in case of emergency. They also allow for minimization of risk.

When deciding on what database to use for any organization, there are a few questions they need to ask themselves: what kind of database do we want? – structured or unstructured, SQL or non-SQL; what is the expected support for such a database? – making a contract with the creators of the database or just using a very popular database that will not be out of support soon; and what are the particular advantages of using a certain database? – some databases allow faster writes or faster reads, other databases allow greater data security through versioning and many other features.

Cloud databases may help with security since they may implement automated backups and scalability. At the same time, they have two inherent disadvantages: the data must travel through an unsecure tunnel, the internet, which has its own solutions such as VPN's (annex 2) and the data is now in the possession of a third party that may or may not be trusted.

Networks and their management are another key component to any organization. Having proper security on any level of the iso/osi model [39] is of upmost importance. Because most software assumes a high level of security within the organization network, incidents may arise because of network mismanagement.

When dealing with sensitive data, not only the access points, but the mode of transfer must also be secure. A secure network is not only maintained and guarded physically; it also must be maintained logically.

When discussing physical security, we can talk about maintaining proper procedures for entering restricted areas and maintaining the equipment.

On the other hand, logical security of a network has many components. There are passive security solutions such as honeypots and canaries. There are active security solutions such as intrusion detection systems.

There can also be layers of security, in the form of DMZ's [40] (annex 2), with ever-more restrictive firewall rules. And Virtual LAN (annex 2) [41] solutions that will lead to a more secure network.

Chapter 4.2 – Database architecture.

This subchapter debates the database architecture for the application. It displays the most important parts of the database for the good functioning of the application.

The database must be able to store certain data about the user's choice of subjects they are interested in. If relational, it must be in a high normal form for data to be stored correctly.

The first three forms and normal form Boyce-Codd are considered the base for a proper database.

For this to happen, a structure such as the one represented in figure 12 must be used.

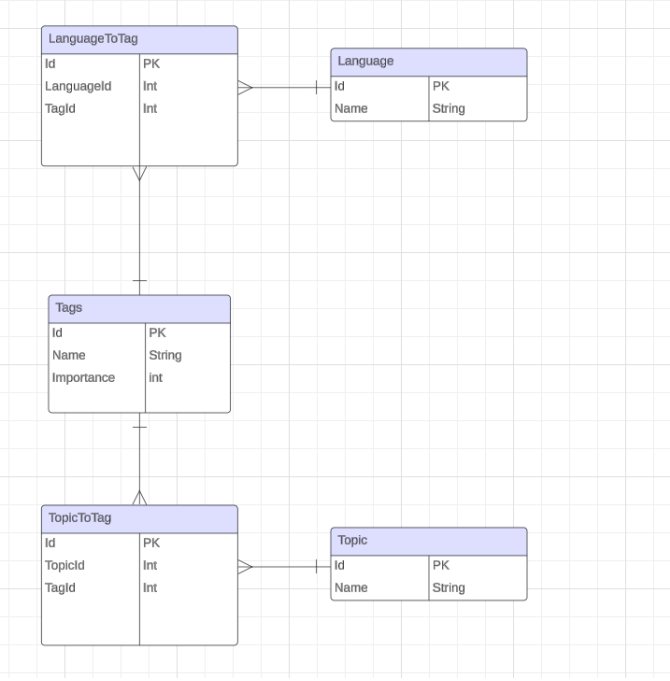

Fig.12 Database ER diagram

The structure depicted shows the linking of languages to security topics through the way of tags. Each language has one or more tags, each topic has one or more tags. The intermediary "LanguageToTag" table allows assignment of multiple tags to a language and multiple languages to a certain tag. The same idea is happening with the table "TopicToTag".

Chapter 4.3 – Server application architecture.

The server application works with the use of a library at serving such requests through the logic inside the "controller classes" and by receiving and sending objects as JSON format in the form described in "object classes". The diagram for this can be found in figure 13.

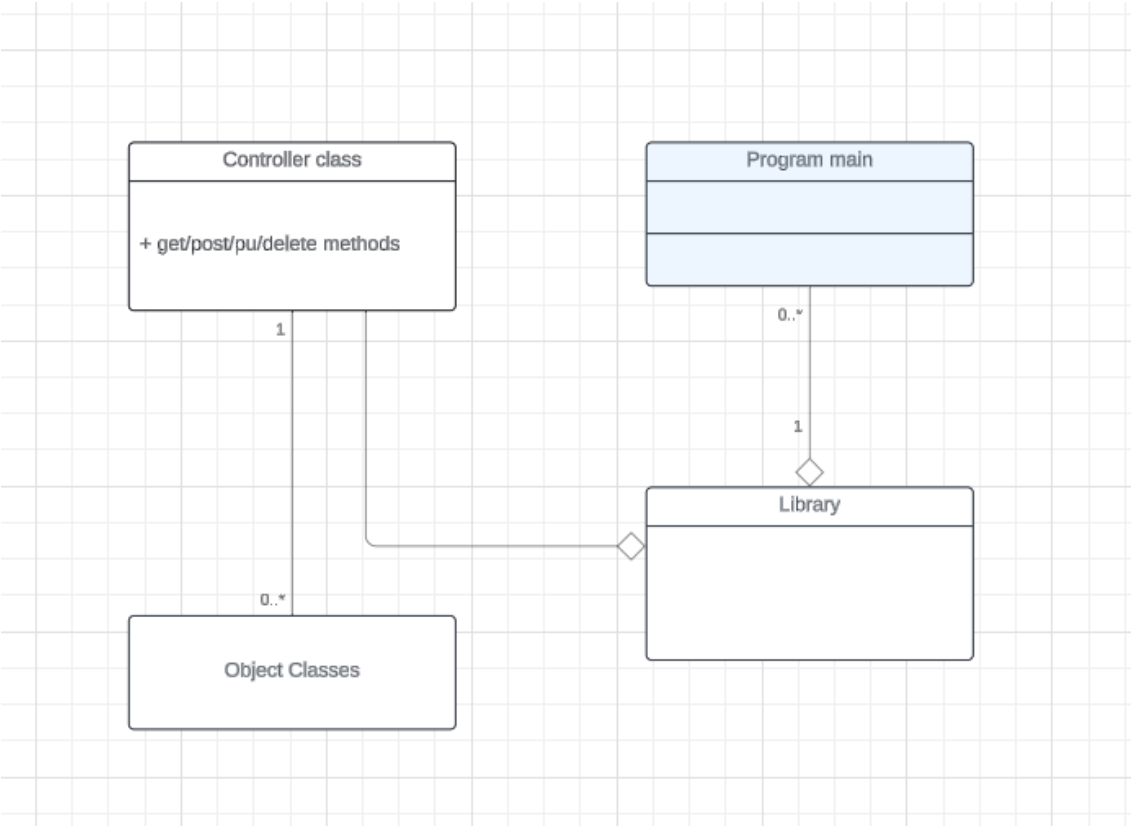

Fig.13 Class diagram

As stated above, the server application must work in tandem with the client application to deliver information. It uses controllers and HTTP (annex 2) methods, alongside JSON (annex 2) formatting to communicate.

The HTTP (annex 2) methods are a standard for information transfer within an API (annex 2):

- Get – to request a resource
- Post – to submit data and may await a response
- Put – to submit data for replacement of existing data
- Delete – to delete specific data

These are some of the most used methods. There are more than those four, but for the purposes of this application, these are all the methods used.

Chapter 4.4 – Client application architecture.

The client application is the way the developer users interact first with the solution application. After these initial interactions, the developer user may be interested in inspecting the code of the application at the API (annex 2) or even the database level.

Below are the diagrams and descriptions of the two most important interactions with the client application.

The use case diagram for the language choosing and the receiving of a vulnerability list can be seen in figure 14.

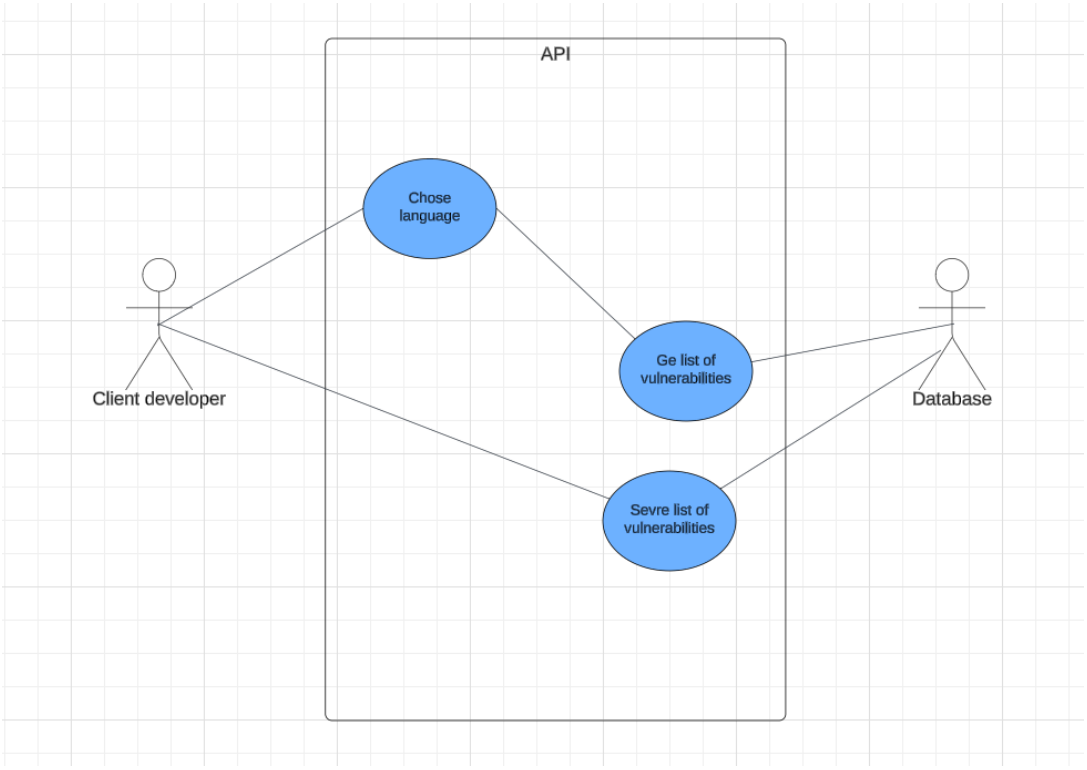

Fig.14 Use case diagram

This initial interaction allows the user to choose their preferred programming language and receive a list of security issues related to that specific language. This is the main purpose of this application, tailoring a list of security issues based on the developer user.

After this interaction, in figure 15 there is a flow chart describing the intended behavior of a user.

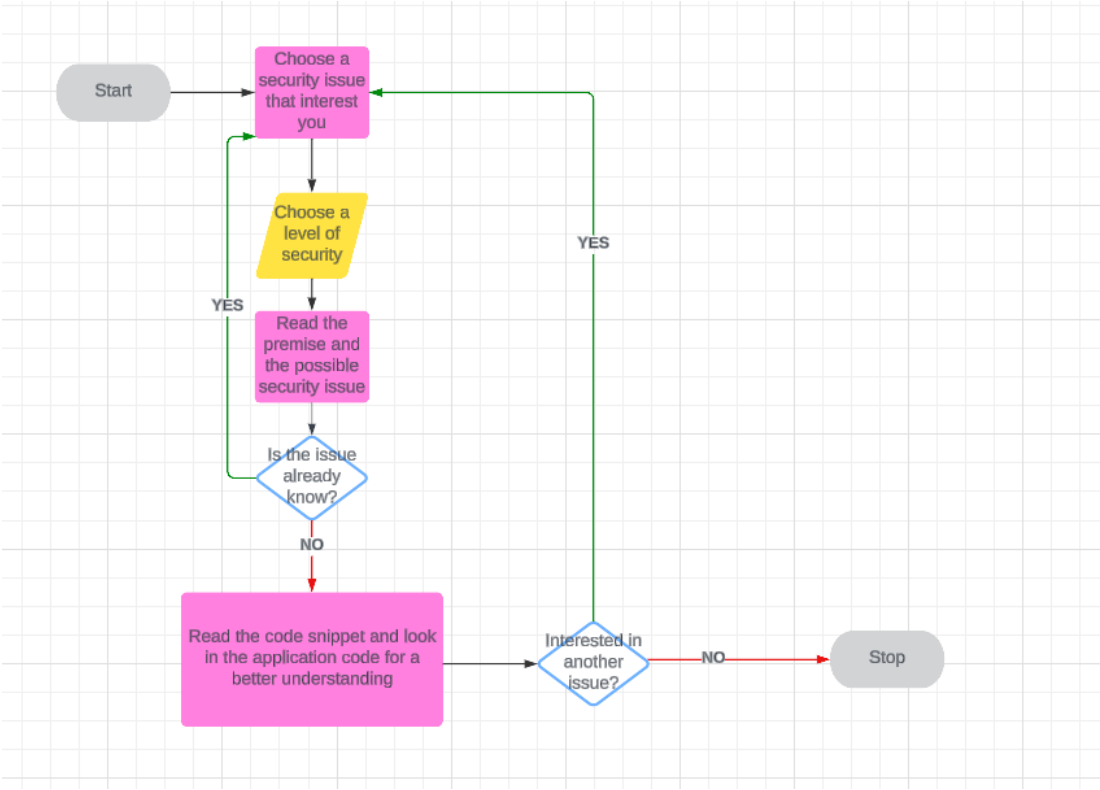

Fig.15 Flow chart diagram

The user is now faced with a list of security issues that have multiple levels of security. They may now select any issue and follow along with the explanation of the issue, the possible problematic code and the possible solution or solutions to it.

Chapter 5 – Application implementation.

This chapter tackles the implementation details of the application.

The application is vulnerable to attacks and its intended use is to be installed and run locally. It is intended as an internal tool for training developers not only in financial institutions, but it may be used in any domain that deals with sensitive data and mandatory uptime.

The application presents the user developer with a curated list of possible security related issues that may arise when using a certain language. The user will then be presented with a list of issues that have levels of security so as for him to understand how the issue may occur and how to deal with it at any level. The application, where possible, makes references to its own code and the user developer is encouraged to look at what is going on "under the hood", in the application client code and API (annex 2) server code.

Chapter 5.1 – Server application implementation.

The server application is created without regard to security, since the requirements do not specify it. With that in mind, the server application must provide data for the list of languages in the client application, their tags and the list of topics.

This is done by API (annex 2) calls. The calls are made by the client application and forwarded to the database through sqlite driver package, as in figure 16.

```
using (var connection = new SqliteConnection(DatabaseConnString))
{
    connection.Open();
    var command = connection.CreateCommand();
    command.CommandText = "SELECT * FROM Language";
    List<LanguageWithTag> list = new List<LanguageWithTag>();
    using (var reader = command.ExecuteReader())
    {
        while (reader.Read())
        {
            var id = Int32.Parse(reader.GetString(0));
            var name = reader.GetString(1);
            LanguageWithTag a = new LanguageWithTag(id, name);
            list.Add(a);
        }
    }


    foreach (var item in list)
    {
        var command2 = connection.CreateCommand();
        command2.CommandText = "SELECT * FROM LanguageToTag where LanguageId=" + item.id.ToString();
        List<int> listOfTagId = new List<int>();

        using (var reader2 = command2.ExecuteReader())
        {
            while (reader2.Read())
            {
                var tagId = Int32.Parse(reader2.GetString(2));
                listOfTagId.Add(tagId);
            }
        }
        List<Tag> listOfTag = new List<Tag>();

        foreach (var tagId in listOfTagId)
        {
            var command3 = connection.CreateCommand();
            command3.CommandText = "SELECT * FROM Tag where Id=" + tagId;
            using (var reader3 = command3.ExecuteReader())
            {
                while (reader3.Read())
                {
                    var id = Int32.Parse(reader3.GetString(0));
                    var name = reader3.GetString(1);
                    Tag a = new Tag(id, name);
                    listOfTag.Add(a);
                }
            }
        }
        item.tags = listOfTag;
    }
    return list;
```

Fig.16 API call for getting list of languages with corresponding tags

The returned data types are defined in the database and in classes such as the one presented in figure 17.

```
8 references
public class LanguageWithTag
{
    4 references
    public int id { get; set; }
    3 references
    public string name { get; set; }
    4 references
    public List<Tag> tags { get; set; }

    0 references
    public LanguageWithTag()
    {
        this.id = 0;
        this.name = "";
        this.tags = new List<Tag>();


    }
    1 reference
    public LanguageWithTag(int id, string name)
    {
        this.id = id;
        this.name = name;
        this.tags = new List<Tag>();
    }

    0 references
    public LanguageWithTag(int id, string name, List<Tag> tags)
    {
        this.id = id;
        this.name = name;
        this.tags = tags;
    }
}
```

Fig.17 “LanguageWIthTag” class

Chapter 5.2 – Client application implementation.

The client application is the way the developer users interact first with the solution application.

The client application being developed by using react Material UI [36] components and custom CSS, looks quite standard and can be customized deeply. The look of the application can be extended and used in the entire application through CSS (annex 2) tags that affect classes or id’s.

The look of the application has been made using CSS (annex 2) and can be modified. A screenshot of it can be found in figure 18.

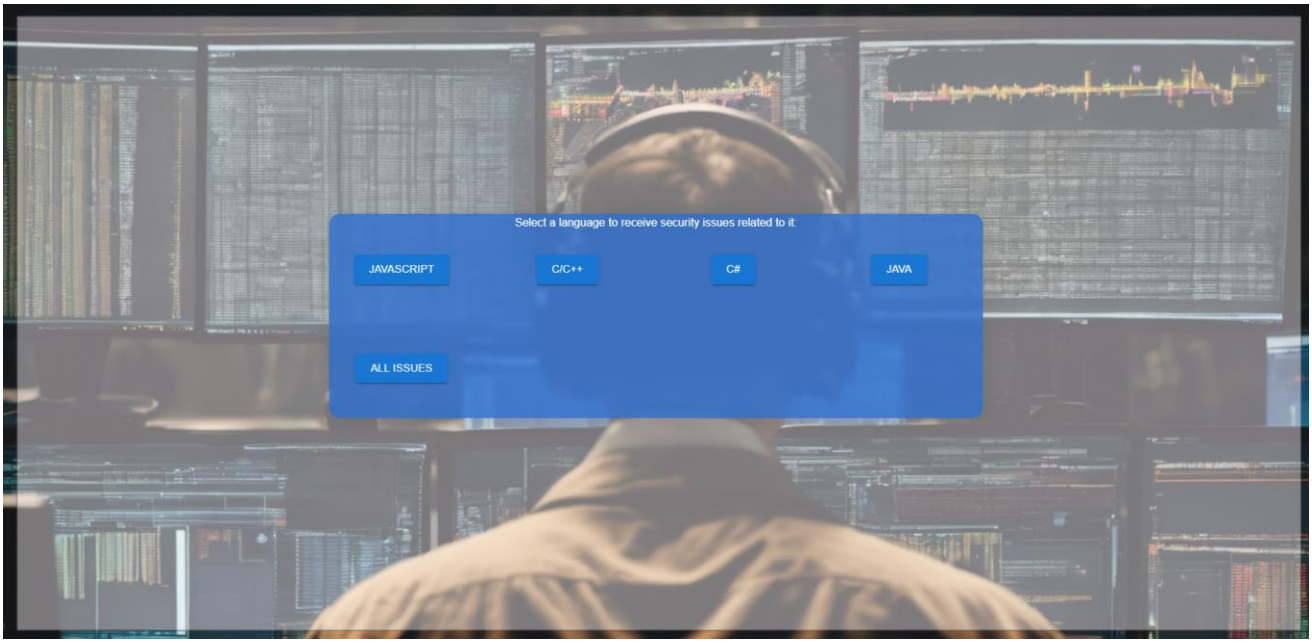

Fig.18 Application selection of language used

Chapter 6 – Conclusion

For the purposes of training developers and bringing awareness, the application is a great tool for internal usage in not only a financial institution but also for any business that is interested in data privacy.

With that in mind, the application can be improved in some respects or changed to fit other acceptance criteria.

One such criteria may be the front-end client list ordering may be improved by adding ordering criteria that can be controlled by the developer user.

Another one can be extending the application from internal tooling to a standalone learning platform by making its API (annex 2) endpoints secure, adding account support, administration pages, statistics and including downloadable projects that represent the current "levels" in security, so as for it not to lose the purpose of developers being able to analyze the secure or insecure code.

As an addition to the initial study, a study can be made on the users of the application, the results being compared to the developers that never interacted with such a system. For this to be viable, the population studied should comprise developers from urban developed countries that used or did not use the application described above or a similar application.

Bibliography

[1] Meta, "React," React, 01 01 2024. [Online]. Available: https://react.dev/. [Accessed 31 05 2024].

[2] Microsoft, ".NET," Microsoft, 01 01 2024. [Online]. Available: https://dotnet.microsoft.com/en-us/download. [Accessed 05 06 2024].

[3] ISO/IEC, "Introducing JSON," ISO/IEC 21778, 01 01 2017. [Online]. Available: https://www.iso.org/standard/71616.html. [Accessed 05 06 2024].

[4] Mozilla, "HTML: HyperText Markup Language," Mozilla, 01 01 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML. [Accessed 05 06 2024].

[5] Mozilla, "CSS: Cascading Style Sheets," Mozilla, 01 01 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS. [Accessed 05 06 2024].

[6] Mozilla, "JavaScript," Mozilla, 01 01 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Accessed 05 06 2024].

[7] Sqlite, "Sqlite," Sqlite, 15 04 2024. [Online]. Available: https://sqlite.org/. [Accessed 05 06 2024].

[8] USA Gov, "financial-institution-definition," 01 01 2024. [Online]. Available: https://www.fincen.gov/financial-institution-definition. [Accessed 1 1 2024].

[9] J. H. M. B. D. G. Christian W. Probst, "Insider Threats in Cyber Security," Springer, Lyngby, Denmark, 2010.

[10] D. B. Johnson, "Tesla says former employees leaked thousands of personal records to German news outlet," SC Media, 2023.

[11] IBM, "IBM X-Force Threat Intelligence Index 2024," IBM, 1 1 2024. [Online]. Available: https://www.ibm.com/account/reg/us-en/signup?formid=urx-52629. [Accessed 25 2 2024].

[12] O. Y. I. K. M. M. I. &. N. V. Shulha, "Banking information resource cybersecurity system modeling.," Journal of Open Innovation: Technology, Market, and Complexity 8.2, Kyiv, Ukraine, 2022.

[13] D. F. M. M. A. R. P. M. Biswajit Panja, "Cybersecurity in Banking and Financial Sector: Security Analysis of a Mobile Banking Application," IEEE, Flint, MI, USA; Ypsilanti, MI, USA, 2013.

[14] A. I. A.-A. a. S. A. Al-Bassam, "Assessing The Factors of Cybersecurity, Awareness in the Banking Sector," -, Riffa, Bahrain, 2021.

[15] C. Grandjean, "Bank Robberies and Physical Security in Switzerland," Butterworth Publishers, Lausanne, Switzerland, 1990.

[16] R. V. R. L. Y. B. T. S. F. Mohd Khairul Affendy Ahmad, "Security Issues on Banking Systems," Universiti Malaysia Sabah, Kota Kinabalu, Malaysia, 2010.

[17] L. J. Fennelly, "Effective Physical Security," Todd Green, Amsterdam, Holand, 2017.

[18] S. Angel, "Discouraging crime through city planning," University of California Press, Berkeley, California, 1968.

[19] A. Y. Angus Wong, Network Infrastructure Security, Hong Kong, PR, China: Springer, 2009.

[20] Z. O. F. S. O. A. O. B. Ogunwobi, "Evaluation of Computer and Network Security Strategies: A Case study of Nigerian Banks," CoRI, Ibadan, Nigeria, 2016.

[21] Cybersecurity Technical Report, "Network Infrastructure Security Guide," National Security Agency, United States, 2023.

[22] K. K. R. P. T. S. Jacob Haislip, "The economic cost of cybersecurity breaches: A broad-based analysis," Workshop on the economics of information security (WEIS). Vol. 9., New York, New York, USA, 2019.

[23] A. a. C. Database Security—Concepts, "Elisa Bertino, Ravi Sandhu," IEEE, 2005.

[24] N. G.-O. Y. G. E. G. J. A. Lior Okman, "Security Issues in NoSQL Databases," IEEE, Changsha, China, 2012.

[25] A. O. a. A. P. G. Blinowski, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," IEEE Access, vol. 10, 2022.

[26] S. V. Lakshmi Iyer, "Cyber Security in API Economy – Issues and Challenges," INCCOCE, Bengaluru, India, 2016.

[27] N. Madden, API Security in Action, Manning, 2020.

[28] M. K. Bond, "Understanding Security APIs," University of Cambridge, Cambridge, UK, 2004.

[29] A. B. K. M. A. C. M. &. D. R. Jibril, "Customers' perception of cybersecurity threats toward e-banking adoption and retention: A conceptual study. In ICCWS 2020 15th International Conference on Cyber Warfare and Security (Vol. 270)," Academic Conferences and publishing limited, Zlin, Czech Republic, 2020.

[30] A. K. Singha, "Risk Assessment of Computer Network Security in Banks," IJRDO Journal, Delhi, India, 2017.

[31] Microsoft, "GitHub Copilot in VS Code," Microsoft, 05 02 2024. [Online]. Available: https://code.visualstudio.com/docs/copilot/overview. [Accessed 26 05 2025].

[32] E. A. ANGWAOMAODOKO, " The Re-examination of the Dangers and Implications of Artificial Intelligence for the Future of Scholarship and Learning," Path of Science, 2023.

[33] DB Browser for SQLite, "DB Browser for SQLite," DB Browser for SQLite, 01 01 2014. [Online]. Available: https://sqlitebrowser.org/. [Accessed 04 06 2024].

[34] Microsoft, "Windows," Microsoft, 01 01 2019. [Online]. Available: https://www.microsoft.com/en-us/windows. [Accessed 04 06 2024].

[35] Microsoft, "Visual Studio Microsoft," Microsoft, 10 05 2022. [Online]. Available: https://visualstudio.microsoft.com/. [Accessed 31 05 2024].

[36] Meta, "Material UI - Overview," Meta, 01 01 2024. [Online]. Available: https://mui.com/material-ui/getting-started/. [Accessed 06 06 2024].

[37] OpenJS Funcation, "Express," OpenJS Funcation, 01 01 2017. [Online]. Available: https://expressjs.com/. [Accessed 01 06 2024].

[38] Microsoft, "Visual Studio Code," VS Code Microsoft, 10 05 2019. [Online]. Available: https://code.visualstudio.com/learn. [Accessed 31 05 2024].

[39] International Organization for Standardization, ISO/IEC 7498-1:1994, Geneva: International Organization for Standardization, 1994.

[40] Department of Homeland Security USA, "Archive of "Control System Security DMZ"," Department of Homeland Security USA, 9 May 2020. [Online]. Available: https://web.archive.org/web/20200609134629/https://www.us-cert.gov/ics/Control_System_Security_DMZ-Definition.html. [Accessed 2 February 2024].

[41] IEEE, "802.1Q-1998 - IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks," IEEE, 1 1 1998. [Online]. Available: https://ieeexplore.ieee.org/document/753056. [Accessed 24 2 2024].

[42] I. S. P. S. Erez Yallon, "OWASP API Security Top 10," 2023. [Online]. Available: https://owasp.org/API-Security/editions/2023/en/0x11-t10/. [Accessed 13 01 2024].

Annex 1 – Figure list

Annex 2 – Acronym list

IT	Information Technology
API	Application Programming Interface
DB	Data Base
TCP	Transmission Control Protocol
LAN	Local Area Network
SQL	Structured Query Language (database type/language)
DMZ	De-Militarized Zone
VPN	Virtual Private Network
AI	Artificial Intelligence
HTTP	HyperText Transfer Protocol
ATM	Automated Teller Machine
JSON	JavaScript Object Notation
POS	Point Of Sale