TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Igor Podgainõi 164754IAPB

IMPLEMENTING AN INTEGRATION COMPONENT FOR A FINANCIAL PERFORMANCE MANAGEMENT SYSTEM AS A MICROSOFT OFFICE 2019 OR 365 "ADD-IN" WITH THE USE OF MODERN TECHNOLOGIES

Bachelor's thesis

Supervisor: Deniss Kumlander
PhD

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Igor Podgainõi 164754IAPB

FINANTSTULEMUSLIKKUSE JUHTIMISSÜSTEEMILE INTEGRATSIOONIKOMPONENDI ARENDAMINE KASUTADES MICROSOFT OFFICE 2019 VÕI 365 "ADD-IN" KAASAEGSETE TEHNOLOOGIATE ABIL

bakalaureusetöö

Juhendaja: Deniss Kumlander
Doktorikraad

Tallinn 2019

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Igor Podgainõi

22.05.2019

Abstract

The goal of this thesis is the development of an integration component for an existing financial performance management system as an "add-in" for Microsoft Office. It may be necessary for companies which perform accounting and financial consolidation using Microsoft Office, but do not want to convert existing documents into a format used by internal software for performance management. As a result, financial performance utilities can be worked with directly inside an Office document, which may be more convenient for employees. The thesis focuses on Excel, an application that is a part of the Microsoft Office suite used for manipulating numbers.

This thesis describes different technologies and their appropriacy in relation to the project so that the development can be completed.

The result of this thesis is a finished Microsoft Excel "add-in", which is connected to the company data warehouse using a microservice and is able to place accounting data from a database into a spreadsheet document.

This thesis is written in English and is 28 pages long, including 6 chapters, 10 figures.

Annotatsioon

Finantstulemuslikkuse juhtimissüsteemile integratsioonikomponendi arendamine kasutades Microsoft Office 2019 või 365 "add-in" kaasaegsete tehnoloogiate abil

Selle töö eesmärgiks on olemasoleva finantstulemuslikkuse juhtimissüsteemile integratsioonikomponendi arendamine kasutades Microsoft Office "add-in". See võib olla vajalik ettevõtetele, kes teostavad raamatupidamis- ja finantskonsolideerimist Microsoft Office'i abil, kuid ei soovi tulemuslikkuse juhtimiseks muunduda olemasolevad dokumendid sisemise tarkvara formaadile. Selle tulemusena saab finantstulemuslikkuse utiliite kasutada otse Office'i dokumendi sees, mis võib olla töötajatele mugavam. Töö keskendub Excel'ile, mis on Microsoft Office'i komplektist rakendus numbrite manipuleerimiseks.

Selles töös kirjeldatakse erinevaid tehnoloogiaid ja nende asjakohasust seoses projektiga, nii et arengut saaks lõpule viia.

Lõputöö tulemus on valmis Microsoft Excel'i "add-in", mis on ühendatud ettevõtte andmehoidlaga mikroserveri abil, ning on võimeline panna arvestusandmeid andmebaasist arvutustabeli dokumendile.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 28 leheküljel, 6 peatükki ja 10 joonist.

List of abbreviations and terms

API	Application Programming Interface
C#	C# programming language
CA	Certification Authority
CFO	Chief Financial Officer
CORS	Cross-Origin Resource Sharing
CSS	Cascading Style Sheets
CSV	Comma-Separated Values file format
DNS	Domain Name System
FPM	Financial Performance Management
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IAT	Issued at Claim field
IP	Internet Protocol
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
JWT	JSON Web Token
LSTM	Long Short-Term Memory neural network
NPM	Node Package Manager
PWA	Progressive Web Application
RNN	Recurrent Neural Network
SQL	Structured Query Language
TLS	Transport Layer Security
URL	Uniform Resource Locator
VBA	Visual Basic for Applications
XLS	Excel Spreadsheet file format
XLSX	Microsoft Office Open XML Format Spreadsheet file format

Table of contents

List of figures

1 Introduction

A substantial amount of companies and enterprises nowadays use a specific set of applications in order to organise their internal workflow. Such a selection of programs is commonly known as an "office suite". Office suites are designed to transfer work done typically in an office from paper-based materials to electronic-based ones while providing compatibility with legacy mediums and increasing workplace productivity [1].

One of the most popular office suites today is Microsoft Office. It is being developed by Microsoft Corporation since 1990 [2]. The newest version of the suite going by the names Office 2019 and Office 365 [3] contains applications such as Word, Excel, PowerPoint, OneNote, Skype for Business and many others. For example, Microsoft Word is an advanced text editor or "word processor", with support for text and document formatting, font styling, tables and is designed to let customers replace regular paper-based documents. Microsoft PowerPoint is designed to be used for creating presentations, which consist of slides that are grouped together. Microsoft Excel can be employed for accounting purposes and manipulating small to large amounts of numerical data. These three applications are the most essential in the Microsoft Office suite because they facilitate the typical processes that are carried out in the office.

Excel makes it simpler for accountants or Chief Financial Officers (CFOs), as opposed to manual calculation, to aggregate and consolidate numbers, because it provides many built-in tools that automate such processes. It may also be possible that a company does not have any specialised accounting software due to it being expensive, while Excel would generally be available. Likewise, Excel provides mechanisms to export data to plain text or CSV files [4], which may be the format used by enterprise performance management applications to process large amounts of data. Finally, spreadsheet documents using the native XLS(X) file format are commonly used for sharing purposes and can be imported to other office suites, such as Google Docs [5]. Therefore, it is useful to deploy an Excel "add-in" to be used by the company employees, so that they can retrieve data directly from the company's data warehouse and complete their work. This data may consist of information such as budget, revenue or spendings, as well as the time and date of financial transactions. Then they can make use of existing utilities to complete the accounting processes.

During the course of the thesis, I worked with a representative of a company that faces the posed problem, and was willing to identify the important features necessary to complete the development of the Excel "add-in".

1.1 Initial analysis

Before the development of the integration component began, I analysed the needs of the company. First of all, it required a possibility to create custom functions to be used inside documents to return the total amount of money from a financial account during a specified time span. Microsoft Excel already contains functions related to statistical work [6], so when integrating additional functionality, the syntax can be made similar to them for the sake of simplicity. Microsoft is providing developers with the Application Programming Interface (API) and documentation to do exactly that [7]. I used this API because it is modern and more actively developed, than the old one [8].

To actually transfer the data from the data warehouse into the "add-in", a microservice was created as an intermediary. The goal of this microservice is to fetch information from the data warehouse's database, convert it into an easy-to-process format, and then pass the data to the "add-in" for an Excel custom function. Such a technique has several benefits. Unlike the direct use of database API, there is no limitation as to what database software should be used by the "add-in". This is an advantage, because the database structure may be modified in the future, while the interface of the microservice is going to stay the same. Since I did not have access to the real internal database of the company I worked with, I created my own with a "mocked" structure. In case there is a need to change the structure of the database, only the database-related programming code within the microservice will have to be modified to account for the changes, while the "add-in" in this case will keep using the same routines.

The second requirement for the "add-in" is the ability to build a detailed report, consisting of financial data taken from the database. I realised that in this case custom functions would not be fit for this task, because they only directly apply to one spreadsheet cell, while the database contains information spanning across many columns and rows, all of which would need to be placed into the report.

For this reason, I chose a different API [9], which provides ways to build a Graphical User Interface (GUI) to input required data using the keyboard, and afterwards place the produced output data into the spreadsheet without using the Excel function syntax. Even though this method of input does not provide the ability to dynamically refresh columns and rows with updated information over time, it is actually not required here, because reports are meant to be static and unaffected by any subsequent updates. With this API I was able to allow the user to type filtering data as an input, and pass it to the microservice to generate only the output data that is necessary at the moment. After this, the "add-in" would receive the data and using the Office API, it would be inserted into a separate worksheet inside the document as static text. As an addition, a PivotTable is also automatically generated. PivotTables are external dynamic table structures created by Microsoft for Office Excel, and the contents of PivotTables can be modified by users [10].

Another requested feature is to be able to transfer information from Excel cells into the "add-in" for further processing. While I did not have a clear indication of what actions need to be taken with the received information, I chose to implement a method of dynamically inserting the primary custom function into the document. Instead of specifying the function parameters manually, they would be generated as cell references based on the data was selected by the user using the mouse before the generation occurs. This feature aims to make it easier to change parameters after generation, since their values, in this case, are located within the spreadsheet itself, unlike the typical method, where they would be stored in the function's parameter list.

Finally, a specific custom function to enable financial forecasting was optionally requested. Because TensorFlow is a popular and applicable machine learning framework [11] that is also available for JavaScript, I chose it for this purpose.

2 Data warehouse microservice development

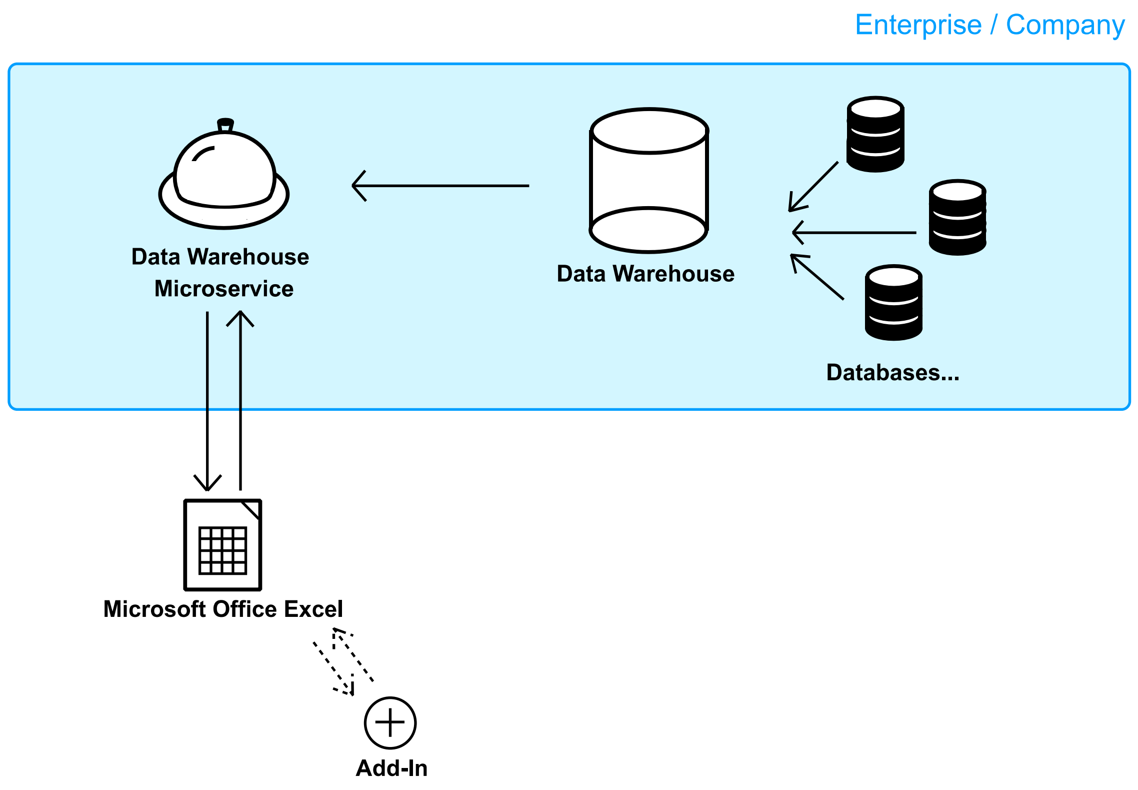


Figure 1. Diagram of data flow between the server and the client

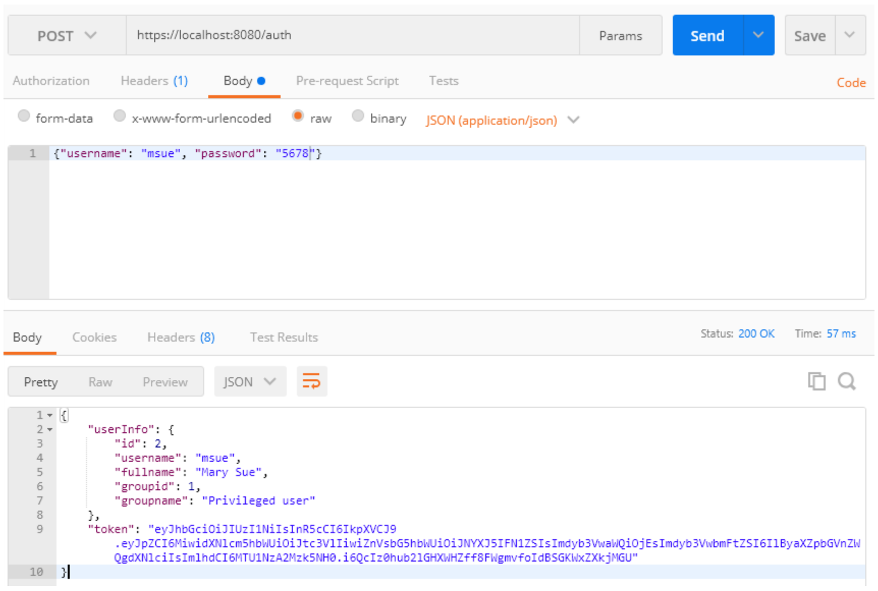
The data warehouse microservice is an additional layer with a stateless architecture, exposing its APIs to be used by the "add-in" to exchange data with the database. It utilises the Hypertext Transfer Protocol/Secure (HTTP/S) protocols for client communication and is written in modern JavaScript based on the ECMAScript 2018 standard using Node.js version 10.15.3 [12]. Because Node.js provides many embeddable packages for use within projects using the NPM package manager, including the Express package [13] for making lightweight web servers, the process of creating the data warehouse microservice has been greatly optimised. A major benefit to Express is the ability to execute code after the HTTP headers and data have already been received by the microservice, but before they reach the specified endpoint. Equivalently, it is also possible to alter data in the other direction after the endpoint function has completed its work, but before the reply is sent back to the client. This way, request headers can be "prepared" for further processing. Editing headers is necessary, because of several "add-in" limitations, such as Simple Cross-Origin Resource Sharing (CORS) [14].

To fetch data from the SQLite 3 database, an NPM package named "better-sqlite3" [15] was used. The database is opened in "read-only" mode to prevent accidental writes during testing.

2.1 User authentication and authorisation

Since the microservice uses the HTTP and HTTPS protocols to communicate with clients, it is normally reachable from any device in the same local network or possibly even the Internet. Therefore, to protect the microservice from potential attacks, it is a good idea to require authorisation to be able to access its APIs.

The authorisation system for corporate users should include privilege separation, since there may be parts of the API that are supposed to be accessible only by certain groups of people and not others. Another feature, which can be considered "nice-to-have" but not necessarily a requirement, is static token generation. Tokens are pieces of information that are used to identify a specific log-in session. Since HTTP(S) requests are stateless by nature, meaning they do not directly contain any relational information about the previous requests made, it could help to minimise the need to keep track of logged in users on the server side.



```
https://localhost:8080/auth
```

```
{"username": "msue", "password": "5678"}
```

Status: 200 OK Time: 57 ms

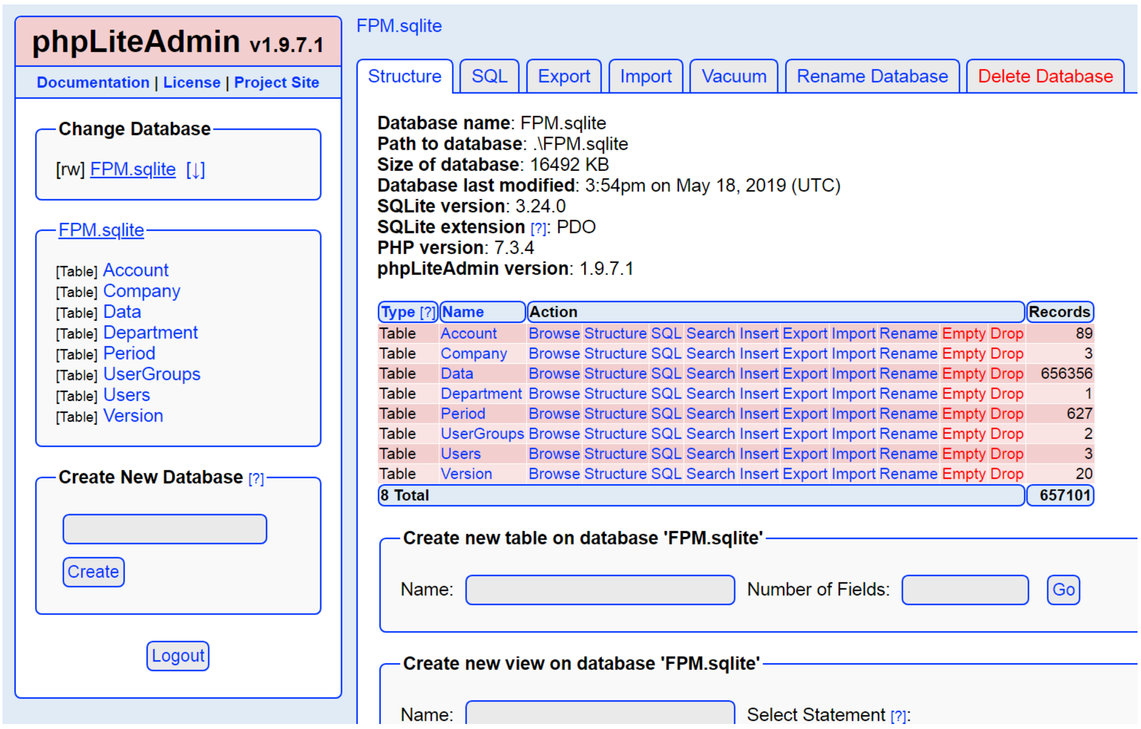
```
{
    "userInfo": {
        "id": 2,
        "username": "msue",
        "fullname": "Mary Sue",
        "groupid": 1,
        "groupname": "Privileged user"
    },
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
        .eyJpZCI6MiwidXNlcm5hbWUiOiJtc3VlIiwiZnVsbG5hbWUiOiJNYXJ5IFN1ZSIsImdyb3VwaWQiOjEsImdyb3VwbmFtZSI6IlByaXZpbGVnZW
        QgdXNlciIsImlhdCI6MTU1NzA2Mzk5NH0.i6QcIz0hub21GHXWHZff8FWgmvfoIdBSGKWxZXkjMGU"
}
```

Figure 2. JSON Web Token example representation in Postman

For these reasons, I chose JSON Web Token (JWT) as the system to be used for authorisation and authenticating users. The main and most important advantage I consider of JWT is that the tokens are automatically encrypted using a secure algorithm

(HMAC-SHA256) [16] and a secret value or key, which is never disclosed outside of microservice's own code. This way, as long as the key stays confidential within the software, the integrity of a JWT token can be guaranteed, because a potential attacker will not be able to simply alter it. Another benefit is that JWT tokens contain all the necessary information about a user inside of themselves, such as the user's full name, without the need to make additional queries to the database. The microservice decrypts the token using the same secret key and immediately receives all needed contextual information about the user. This helps to keep tokens stateless as well, however, a potential vulnerability could be utilising expired tokens for new API requests, which can be classified as a "replay attack" [17]. Fortunately, JWT tokens also include a timestamp in the "IAT" field [18], capturing the time at which it was generated. This means the attack can be mitigated by writing only this timestamp to the database in place of a user's session if necessary and disallowing all other tokens that have their timestamp set to a prior moment. In this project, I have not implemented such protective measures, but it is possible to improve the microservice further by adding additional security features like this one.

2.2 Database structure and mock-up



Database name: FPM.sqlite
Path to database: .\FPM.sqlite
Size of database: 16492 KB
Database last modified: 3:54pm on May 18, 2019 (UTC)
SQLite version: 3.24.0
SQLite extension [?]: PDO
PHP version: 7.3.4
phpLiteAdmin version: 1.9.7.1

Type [?]	Name	Action	Records
Table	Account	Browse Structure SQL Search Insert Export Import Rename Empty Drop	89
Table	Company	Browse Structure SQL Search Insert Export Import Rename Empty Drop	3
Table	Data	Browse Structure SQL Search Insert Export Import Rename Empty Drop	656356
Table	Department	Browse Structure SQL Search Insert Export Import Rename Empty Drop	1
Table	Period	Browse Structure SQL Search Insert Export Import Rename Empty Drop	627
Table	UserGroups	Browse Structure SQL Search Insert Export Import Rename Empty Drop	2
Table	Users	Browse Structure SQL Search Insert Export Import Rename Empty Drop	3
Table	Version	Browse Structure SQL Search Insert Export Import Rename Empty Drop	20
8 Total			657101

Create new table on database 'FPM.sqlite'

Name: Number of Fields: Go

Create new view on database 'FPM.sqlite'

Name: Select Statement [?]:

Figure 3. phpLiteAdmin version 1.9.7.1 main interface

For the project's back-end database system, SQLite version 3 was chosen, because it is lightweight yet in many ways compatible with the standard SQL syntax [19]. Each SQLite database is stored in one separate file in any preferred directory [20]. phpLiteAdmin version 1.9.7.1 running on PHP 7 was used to create and manage the database. Because PHP contains the entire SQLite engine in itself, there is no need to install additional software or perform separate SQL backup dumps. For use in Node.js however, an additional package named "better-sqlite3" had to be installed to support and read SQLite files.

The created database represents a hypothetical data warehouse of a company and contains the following tables: Account, Company, Department, Period, Version, UserGroups, Data, Users. The listed tables except the last two are classifier tables, which means they contain descriptional data to be joined with other tables by using a foreign key. The "Data" table contains a list of financial transactions made, with some information referencing classifier tables, and the "Users" table contains a list of all registered users for the service.

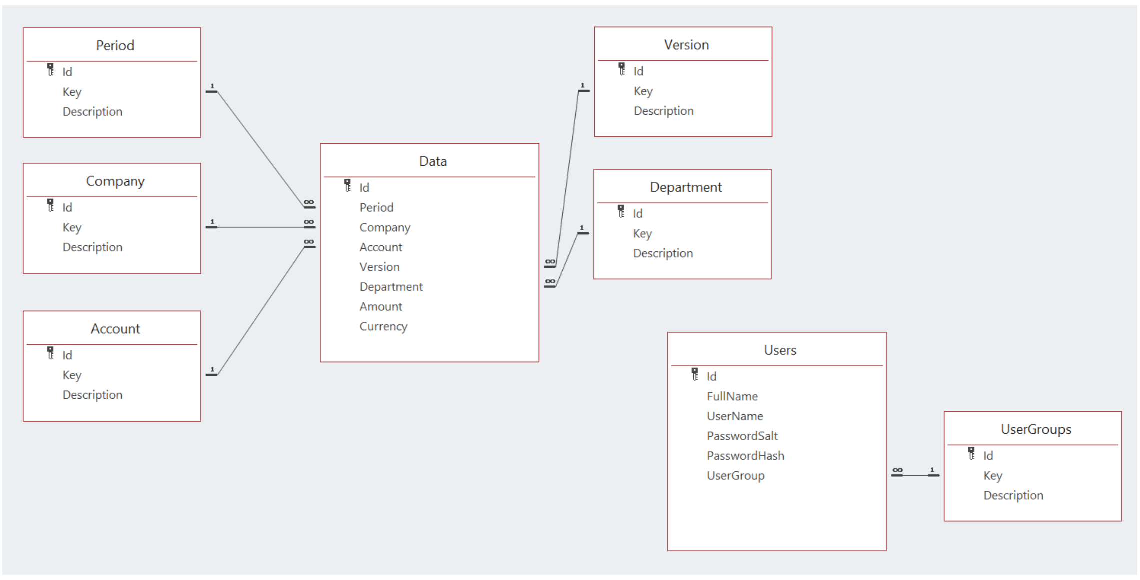


Figure 4. Relationships between database tables in Microsoft Access

Microsoft Access was also used to visualise the relationships between the tables of the "mocked" database.

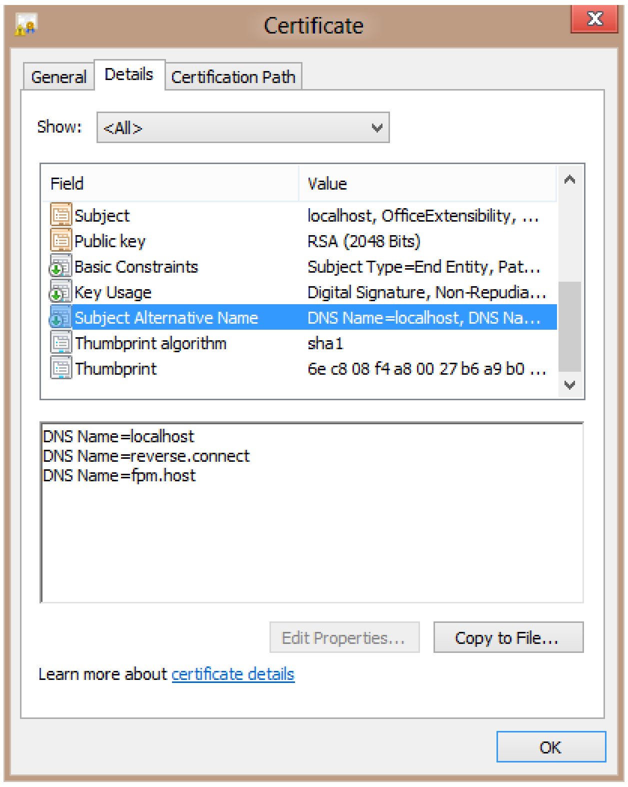
3 "Add-in" development

The creation of the "add-in" itself was made possible by a set of APIs that Microsoft provides for Office developers. The specific APIs I used in the course of this thesis are a relatively new addition because they are designed to be used with a modern form of the JavaScript programming language, unlike the former ones, that were meant to be used with Visual Basic for Applications (VBA) or C# [21].

The first API includes functions related to the Graphical User Interface (to be implemented as a sidebar or task pane), as well as spreadsheet manipulation. This was used to create the main Financial Performance Management (FPM) user interface. The sidebar is a wrapper around Internet Explorer, which displays an HTML page at a predefined Uniform Resource Locator (URL). From this page, JavaScript code can be loaded into memory and afterwards, the Office API can be initialised. This allowed me to design the application the same way as it is typically done when targeting a regular web browser by web developers.

Another API that was used for the "add-in" is a separate custom functions API. At the time of writing this thesis, this API was still in "preview" stage [22], but Microsoft aims to update it and its related documentation from time to time. Since this API is independent, most function calls to the main Office API cannot be performed within it. Besides, it does not utilise Internet Explorer as a backbone, but a custom JavaScript engine, which means it is more functionally limited. To transfer information between the main API and the custom functions API, a class named AsyncStorage is exposed for this purpose. This class provides dictionary-type storage in an asynchronous manner, which is shared between the two APIs. Methods getItem and setItem allow the developer to respectively read and write an element to this internal storage [23].

3.1 API limitations for HTTP requests



Field	Value
Subject	localhost, OfficeExtensibility, ...
Public key	RSA (2048 Bits)
Basic Constraints	Subject Type=End Entity, Pat...
Key Usage	Digital Signature, Non-Repudia...
Subject Alternative Name	DNS Name=localhost, DNS Na...
Thumbprint algorithm	sha1
Thumbprint	6e c8 08 f4 a8 00 27 b6 a9 b0 ...

```
DNS Name=localhost
DNS Name=reverse.connect
DNS Name=fpm.host
```

Figure 5. Opened HTTPS server certificate with DNS names shown

The "add-in" APIs provided by Microsoft require that HTTP requests are only made with Transport Layer Security (TLS) encryption enabled. This type of request is also known as HTTPS, where S stands for "secure", and is a layer on top of regular HTTP [24]. Apart from that, the custom functions API only allows Simple CORS requests, which means that most HTTP headers are denied, with the exception of "Accept", "Accept-Language", "Content-Language" and "Content-Type" [25]. Also, loopback connections using a reserved address such as "localhost" to the same computer are disallowed as well. This means, that for manually testing the "add-in" before deployment into production, self-signed HTTPS certificates and a "trusted root" Certification Authority (CA) must be generated. To make sure that the requests are able to contact a host by DNS name instead of an IP address, all required DNS names must also be added into the certificate body. The project folder for the thesis includes OpenSSL scripts with DNS names "localhost", "reverse.connect" and "fpm.host" attached, which can generate the appropriate files.

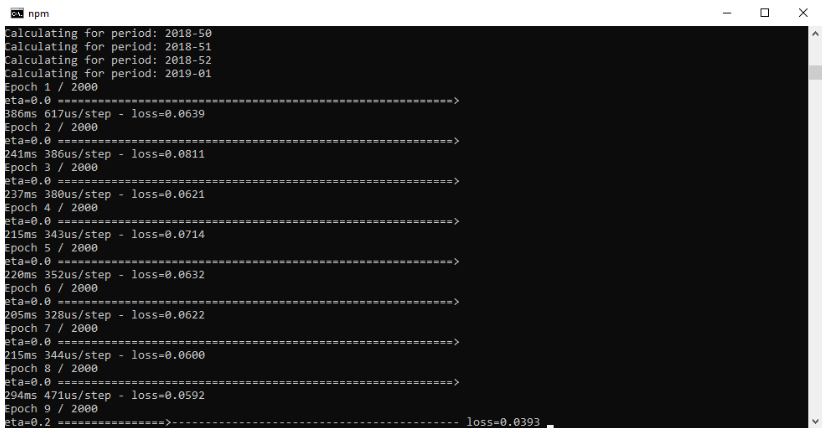
3.2 Currency conversion

To implement currency conversion, two currency-related functions are available in a separate file named "currencyconversion.js". This code is temporary and since it contains fixed currency rates allowing conversion only from Euro and to Euro, it should be replaced in production by developers more familiar with the field of finance. It also utilises the Big.js library to avoid floating point inaccuracies. Currency conversion is used by the "add-in" when letting the user select any supported type of currency in the output.

3.3 Financial forecasting

Using modern frameworks such as TensorFlow, it can be trivial to define and train machine learning models. In this project, I used the TensorFlow JavaScript API to enable financial forecasting functionality within the "add-in". Unfortunately, the JavaScript variant of TensorFlow named TensorFlow.js does not yet fully support back-ends other than Node.js and Google Chrome's underlying browser engine [26], which means that I was not able to embed TensorFlow to the "add-in" itself for forecasting purposes, so I had to import its functionality to the data warehouse microservice instead, which works on top of Node.js.

For testing the forecasting abilities, I made use of a variant of Recurrent Neural Networks (RNN) called Long Short-Term Memory (LSTM). Recurrent Neural Networks are usually used for sequential numerical data and are designed to predict future outcomes, based on all trained data from the past at once [27].



```
Calculating for period: 2018-50
Calculating for period: 2018-51
Calculating for period: 2018-52
Calculating for period: 2019-01
Epoch 1 / 2000
eta=0.0 ===========================================================>
386ms 617us/step - loss=0.0639
Epoch 2 / 2000
eta=0.0 ===========================================================>
241ms 386us/step - loss=0.0811
Epoch 3 / 2000
eta=0.0 ===========================================================>
237ms 380us/step - loss=0.0621
Epoch 4 / 2000
eta=0.0 ===========================================================>
215ms 343us/step - loss=0.0714
Epoch 5 / 2000
eta=0.0 ===========================================================>
220ms 352us/step - loss=0.0632
Epoch 6 / 2000
eta=0.0 ===========================================================>
205ms 328us/step - loss=0.0622
Epoch 7 / 2000
eta=0.0 ===========================================================>
215ms 344us/step - loss=0.0600
Epoch 8 / 2000
eta=0.0 ===========================================================>
294ms 471us/step - loss=0.0592
Epoch 9 / 2000
eta=0.2 ===============>-------------------------------------------- loss=0.0393
```

Figure 6. Model training in the "forecasting generator" with the help of TensorFlow

Training data and resulting models are generated by a separate program, which is named "forecasting generator". This application logs in to the data warehouse service using credentials of a specified privileged user and requests all financial data within a time frame denoted by a year and week, and afterwards computes the resulting price per year in Euros. The data is then saved as a local file to speed up the generation process during later launches. Afterwards, the financial data is normalised using a mathematical normalisation function and passed to TensorFlow as it trains the LSTM model for 2000 cycles (epochs). The epoch count, as well as other settings, can, of course, be tweaked in the future by other developers. The resulting model is saved as a local file, along with the information needed to denormalise the predicted values in other programs.

3.4 Implementing custom functions

The two defined custom functions are designed to output only one piece of information to an Excel spreadsheet cell, in both cases containing an amount of money and its respective currency code according to the ISO 4217 standard. The data type of the output is "string", which means it can consist of any textual information. For example, in case of a failed request, an error message is being output instead of the expected value.

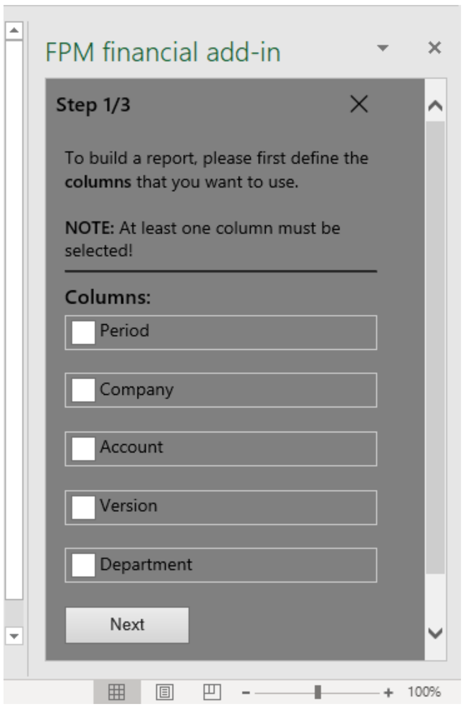
To actually output the value itself, the custom functions first retrieve a previously saved JWT authentication token from AsyncStorage to be used as an HTTP(S) header and then send it to the microservice along with the input parameters. Here, the "Currency" parameter is optional, and if no value for it is specified, it defaults to "EUR" (Euros).

For the first function, the microservice sends back an array of values to the client that denote either a negative or positive amount of money and/or the currency. Then all the amounts are summed up on the client side with the help of the Big.js library.

The second function behaves the same way, except for actions taken after the data has been received from the microservice. In this case, it is only a single amount in Euros, which does not need to be added up and can be directly converted to another currency using a currency conversion function.

3.5 Sidebar as the main GUI

With the sidebar, a user of the "add-in" can access the main Graphical User Interface, where several modes of operations (labelled as "Function") may be selected. The GUI represents all available functionality that is not available via the custom Excel functions. It is made with the help of HTML, CSS and JavaScript, which are the primary web development technologies.


FPM financial add-in

Step 1/3

To build a report, please first define the **columns** that you want to use.

NOTE: At least one column must be selected!

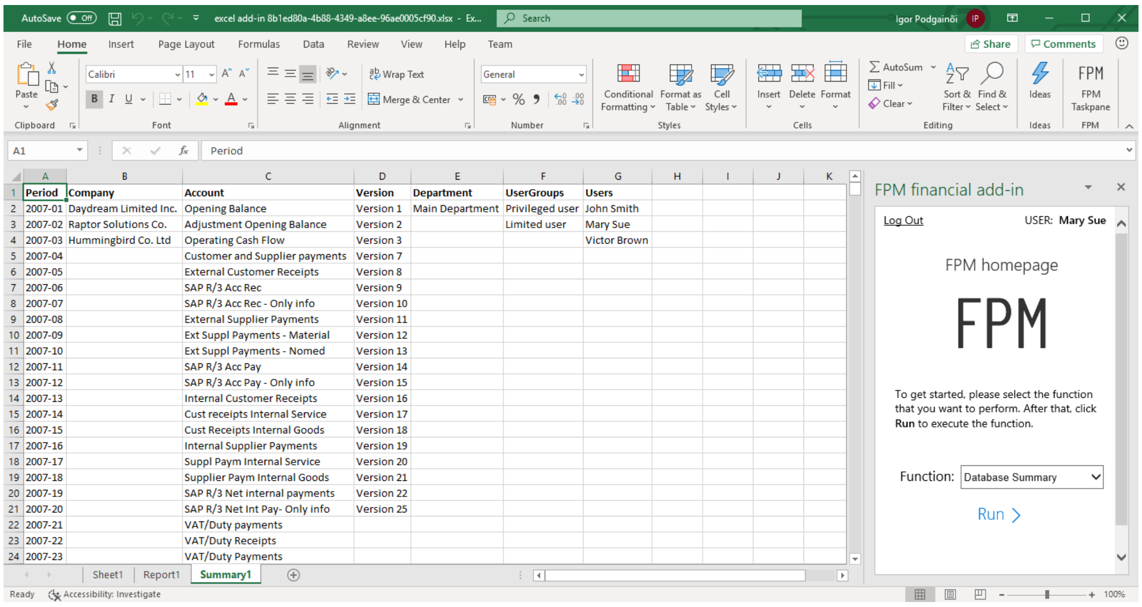
Columns:

- Period
- Company
- Account
- Version
- Department

Figure 7. FPM main interface with "Build Report" mode shown (step 1)

The first mode instructs the user to go through a series of steps and select the desired data to filter by inputting classifier values. If no mistake has been made, a separate worksheet is created, where the output is written to.

The second mode allows the user to automatically insert the main custom function into the spreadsheet, however, the parameters, in this case, are being taken from nearby cells and selected using the mouse.



Period	Company	Account	Version	Department	UserGroups	Users
2007-01	Daydream Limited Inc.	Opening Balance	Version 1	Main Department	Privileged user	John Smith
2007-02	Raptor Solutions Co.	Adjustment Opening Balance	Version 2		Limited user	Mary Sue
2007-03	Hummingbird Co. Ltd	Operating Cash Flow	Version 3			Victor Brown
2007-04		Customer and Supplier payments	Version 7			
2007-05		External Customer Receipts	Version 8			
2007-06		SAP R/3 Acc Rec	Version 9			
2007-07		SAP R/3 Acc Rec - Only info	Version 10			
2007-08		External Supplier Payments	Version 11			
2007-09		Ext Suppl Payments - Material	Version 12			
2007-10		Ext Suppl Payments - Nomed	Version 13			
2007-11		SAP R/3 Acc Pay	Version 14			
2007-12		SAP R/3 Acc Pay - Only info	Version 15			
2007-13		Internal Customer Receipts	Version 16			
2007-14		Cust receipts Internal Service	Version 17			
2007-15		Cust Receipts Internal Goods	Version 18			
2007-16		Internal Supplier Payments	Version 19			
2007-17		Suppl Paym Internal Service	Version 20			
2007-18		Supplier Paym Internal Goods	Version 21			
2007-19		SAP R/3 Net internal payments	Version 22			
2007-20		SAP R/3 Net Int Pay- Only info	Version 25			
2007-21		VAT/Duty payments				
2007-22		VAT/Duty Receipts				
2007-23		VAT/Duty Payments				

FPM financial add-in

Log Out USER: Mary Sue

FPM homepage

FPM

To get started, please select the function that you want to perform. After that, click **Run** to execute the function.

Function: Database Summary

Run >

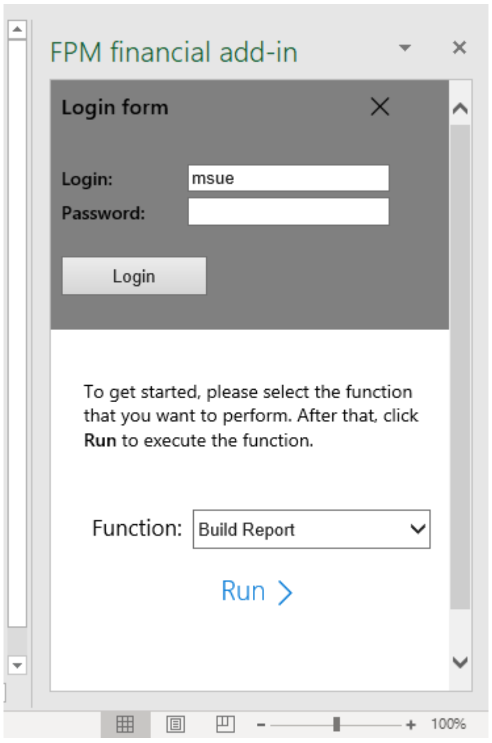
Figure 8. Spreadsheet with summary view

The third and final mode provides a summary of the database's classifiers through a special microservice endpoint and outputs it to a separate worksheet.

4 Application testing

Testing is a crucial part of the development process so that it can be made sure that all prerequisites are still fulfilled after making changes to the code. Both manual testing and unit testing principles are used in the course of the project.

4.1 Using the "add-in"


FPM financial add-in

Login form

Login: msue
Password:

Login

To get started, please select the function that you want to perform. After that, click **Run** to execute the function.

Function: Build Report

Run

Figure 9. Main FPM Graphical User Interface with login form shown

To enable the "add-in", a new document has to be created via NPM scripts in the folder "AddinFinancial". This will open Excel automatically with an extra button that launches the main interface. Before using any of the features provided by the "add-in", a user must first authenticate using his or her corporate credentials. A login form is provided in the sidebar, and upon successful login, the resulting JWT token is saved to AsyncStorage. After that, modes of operation on the main screen of the sidebar are available for usage. Whereas some of them do not need any feedback from the user to complete, others require that either a certain action is taken beforehand, or a GUI form is filled in. If by chance this action was not done properly, an error message is displayed.

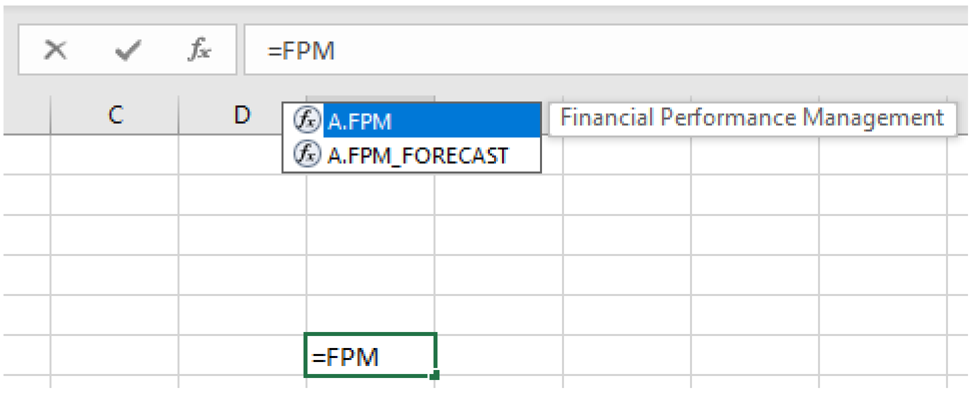


Figure 10. Available custom functions provided by the "add-in"

Custom functions can make use of the saved token as well. There are two custom functions available: "=A.FPM" and "=A.FPM_FORECAST". They both need a correct token to retrieve any data and can be used the same way as standard Excel functions.

After the necessary work is done, the user can log out of the "add-in" by clicking the "Log Out" button, and after this operation, the JWT token is cleared and no longer available.

4.2 Unit testing

To avoid regressions and software bugs in the project's code, it was suitable to employ unit testing during development, in which typical scenarios are being tested at the level of a unit, here specifically a function. A specific unit test succeeds if for a given defined input the output is the same as the expected one. The strategy of unit testing I chose has the characteristics of white-box testing, where apart from testing the input and output of a unit, the correctness of internal variables or states is checked as well [28].

The unit testing program was written in JavaScript, just like the main project. Even though there is an abundance of ready-to-use unit testing frameworks available for JavaScript [29], I required "stubbing" functionality that was not commonly available. In addition to that, I wanted to have the ability to extract separate functions from another file and load them using the "Function" object built into the language. Therefore, I built the unit testing program from scratch. The created application contains a small set of functions that can be used in place of a regular comprehensive testing framework.

Apart from locally verifying source code files, endpoint-level testing over the network is used as well, because the format of endpoint definitions utilised by the Express library within the data warehouse microservice does not reflect standard JavaScript functions.

5 Possible improvements

The "add-in" can be improved further by remaking the Graphical User Interface as a Progressive Web Application (PWA) using frameworks such as React or Vue.js. This way, the modularity of GUI components is increased.

Another enhancement that could be made is the tweaking the financial forecasting functionality, for example by changing the current neural network settings, or using a different one altogether. Additionally, more ways to normalise financial data can be used.

Finally, the support for Google Sheets, a web-based Microsoft Excel alternative and a part of the Google Docs office suite, can be explored as well. The bare minimum implementation for a financial performance management integration component for Google Sheets is already present in the project as "database summary" and "financial forecasting" functions in the "GoogleSheets" folder.

6 Summary

The main goal of the thesis was to make an integration component for a financial performance management system. An analysis was made to determine the correct ways to complete the work. After the completion and testing of the project, I can conclude, that all of the necessary required features have been implemented, and the result is a finished "add-in" for Microsoft Office Excel. In brief, it is able to utilise a microservice to be able to retrieve data from the company's databases seamlessly to the user and modify a spreadsheet document in different ways.

All of the contained features of the project include:

- The ability for the "add-in" to connect to the data warehouse microservice
- The ability for the data warehouse microservice to connect the company's databases
- The ability to use a custom function to fetch and aggregate data from the microservice
- The ability to convert financial values across currencies
- The ability for the user to use a Graphical User Interface via the sidebar
- The ability to create a report on a separate worksheet that displays static filtered data to the user
- The ability to dynamically insert a custom function into the spreadsheet based on user's input
- The ability to predict financial information based on the past data (forecasting)
- The ability for the data warehouse microservice to alter HTTP(S) request headers

- The ability for the user to log into the "add-in" using enterprise credentials
- The ability to store the user's session token locally after a login
- The ability to retrieve information about the user and generation time directly from the token
- The ability to transfer the session token from the "add-in" to the microservice
- The ability to automatically test the project's code for correctness (unit testing)
- The ability to write custom support for Google Sheets, based on the data provided

The finished project directory contains everything necessary to be able to use the "add-in" and develop it further.

References

[1] “Productivity software. Office suite,” 22 5 2019. [Online]. Available: https://en.wikipedia.org/wiki/Productivity_software#Office_suite.

[2] “History of Microsoft Office,” 22 5 2019. [Online]. Available: https://en.wikipedia.org/wiki/History_of_Microsoft_Office.

[3] “Microsoft Office 2019,” 22 5 2019. [Online]. Available: https://en.wikipedia.org/wiki/Microsoft_Office_2019.

[4] “Import or export text (.txt or .csv) files,” 22 5 2019. [Online]. Available: https://support.office.com/en-us/article/import-or-export-text-txt-or-csv-files-5250ac4c-663c-47ce-937b-339e391393ba.

[5] “Import data sets & spreadsheets,” 22 5 2019. [Online]. Available: https://support.google.com/docs/answer/40608.

[6] “Statistical functions (reference),” 22 5 2019. [Online]. Available: https://support.office.com/en-us/article/statistical-functions-reference-624dac86-a375-4435-bc25-76d659719ffd.

[7] “Custom functions parameter options,” 9 5 2019. [Online]. Available: https://docs.microsoft.com/en-us/office/dev/add-ins/excel/custom-functions-parameter-options.

[8] “Create custom functions in Excel,” 22 5 2019. [Online]. Available: https://support.office.com/en-us/article/create-custom-functions-in-excel-2f06c10b-3622-40d6-a1b2-b6748ae8231f.

[9] “Build an Excel task pane add-in,” 2 5 2019. [Online]. Available: https://docs.microsoft.com/en-us/office/dev/add-ins/quickstarts/excel-quickstart-jquery.

[10] “Create a PivotTable to analyze worksheet data,” 22 5 2019. [Online]. Available: https://support.office.com/en-ie/article/create-a-pivottable-to-analyze-worksheet-data-a9a84538-bfe9-40a9-a8e9-f99134456576.

[11] “Why TensorFlow always tops machine learning and artificial intelligence tool surveys,” 23 8 2018. [Online]. Available: https://hub.packtpub.com/tensorflow-always-tops-machine-learning-artificial-intelligence-tool-surveys/.

[12] “Node.js ES2018 Support,” 22 5 2019. [Online]. Available: https://node.green/#ES2018.

[13] “Installing Express,” 22 5 2019. [Online]. Available: http://expressjs.com/en/starter/installing.html#installing.

[14] “Runtime for Excel custom functions,” 8 5 2019. [Online]. Available: https://docs.microsoft.com/en-us/office/dev/add-ins/excel/custom-functions-runtime.

[15] “better-sqlite3 - npm,” 22 5 2019. [Online]. Available: https://www.npmjs.com/package/better-sqlite3.

[16] “Introduction to JSON Web Tokens,” 22 5 2019. [Online]. Available: https://jwt.io/introduction/.

[17] “Use JWT The Right Way!,” 7 10 2014. [Online]. Available: https://stormpath.com/blog/jwt-the-right-way.

[18] “JSON Web Token - Structure,” 22 5 2019. [Online]. Available: https://en.wikipedia.org/wiki/JSON_Web_Token#Structure.

[19] “SQLite - SQL As Understood By SQLite,” 22 5 2019. [Online]. Available: https://www.sqlite.org/lang.html.

[20] “SQLite - Database File Format,” 22 5 2019. [Online]. Available: https://www.sqlite.org/fileformat.html.

[21] “AddIn object (Excel),” 29 3 2019. [Online]. Available: https://docs.microsoft.com/en-us/office/vba/api/excel.addin.

[22] “Create custom functions in Excel (preview),” 1 5 2019. [Online]. Available: https://github.com/OfficeDev/office-js-docs-pr/blob/296f6364b12a794c886e03de80b6d041786e714f/docs/excel/custom-functions-overview.md.

[23] “OfficeRuntime.AsyncStorage,” 1 5 2019. [Online]. Available: https://bit.ly/2JXHVkP.

[24] “HTTPS - Wikipedia,” 22 5 2019. [Online]. Available: https://en.wikipedia.org/wiki/HTTPS.

[25] “Fetch Standard - Headers,” 3 5 2019. [Online]. Available: https://fetch.spec.whatwg.org/#cors-safelisted-request-header.

[26] “Platform and environment | TensorFlow.js,” 22 5 2019. [Online]. Available: https://www.tensorflow.org/js/guide/platform_environment.

[27] “The magic of LSTM neural networks,” 2 2 2018. [Online]. Available: https://medium.com/datathings/the-magic-of-lstm-neural-networks-6775e8b540cd.

[28] “Testing advice. Chapter 9. Software Testing,” 22 5 2019. [Online]. Available: https://www.cs.uct.ac.za/mit_notes/software/htmls/ch09s06.html.

[29] “JavaScript Unit Testing Frameworks,” 22 5 2019. [Online]. Available: https://www.guru99.com/javascript-unit-testing-frameworks.html.