CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Nuclear Sciences and Physical Engineering

Web Application for Registering Finances of Small Businesses

Webová aplikace pro daňovou evidenci drobného podnikatele

Bachelor's Degree Project

Author:	**Martin Čepela**
Supervisor:	**Doc. Ing. Miroslav Virius, CSc.**
Language advisor:	**PaedDr. Eliška Rafajová**
Academic year:	2023/2024

České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská

Katedra: matematiky Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Martin Čepela
Studijní program:	Aplikovaná informatika
Název práce (česky):	Webová aplikace pro daňovou evidenci drobného podnikatele
Název práce (anglicky):	Web application for registering finances of small businesses

Pokyny pro vypracování:

1) Seznamte se s problematikou daňové evidence drobného podnikatele včetně odpovídajících zákonů.
2) Analyzujte požadavky na webovou aplikaci, která umožní drobným podnikatelům vést evidenci příjmů a výdajů na účtě, příjmů a výdajů v hotovosti, dlouhodobého a krátkodobého majetku, pohledávek a dluhů.
3) Navrženou aplikaci implementujte a otestujte.
4) K vytvořené aplikaci vypracujte uživatelskou příručku.

Doporučená literatura:

1) 1. MINISTERSTVO FINANCÍ ČESKÉ REPUBLIKY. Vymezení pojmu drobný, malý a střední podnikatel a postup pro zařazování podnikatelů do jednotlivých kategorií [online] [visited on 2021-12-17]. Available from: https://www.mpo.cz/assets/dokumenty/27300/28024/310086/priloha017.pdf.

2) Zákon o účetnictví (úplné znění). Předpis č. 563/1991 Sb. ve znění platném od 1. 1. 2022.

3) Oliver Leiss, Jasmin Schmidt, PHP v praxi. Praha: Grada Publishing 2010. ISBN 978-80-247-3060-8

Jméno a pracoviště vedoucího bakalářské práce:

Doc. Ing. Miroslav Virius, CSc.
KSI FJFI ČVUT v Praze, Trojanova 13, 120 00 Praha 2

Jméno a pracoviště konzultanta:

Datum zadání bakalářské práce: 31.10.2022

Datum odevzdání bakalářské práce: 2.8.2023

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 31.10.2022

....
garant oboru

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta jaderná a fyzikálně inženýrská
-3-

....
vedoucí katedry

....
děkan

Acknowledgment:

I would like to thank Doc. Ing. Miroslav Virius, CSc. for his expert guidance and express my gratitude to PaedDr. Eliška Rafajová for her language assistance.

Author's declaration:

I declare that this Bachelor's Degree Project is entirely my own work and I have listed all the used sources in the bibliography.

Prague, January 8, 2024 Martin Čepela

Název práce:

Webová aplikace pro daňovou evidenci drobného podnikatele

Autor: Martin Čepela

Obor: Aplikovaná informatika

Druh práce: Bakalářská práce

Vedoucí práce: Doc. Ing. Miroslav Virius, CSc., České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská, Katedra softwarového inženýrství

Abstrakt: Tato bakalářská práce řeší složitost daňové evidence a výzvy, kterým drobní podnikatelé musí čelit. Cílem této bakalářské práce bylo vytvořit přehlednou webovou stránku pro vedení daňové evidence drobného podnikatele. Aplikace je zpracována na základě odpovídajících zákonů pro daňovou evidenci volně přístupných účetnických softwarových aplikací. Vzhled aplikace byl vyřešen pomocí JavaScriptové open-source knihovny Bootstrap. Tímto projektem jsme odpověděli na absenci přehledných specializovaných nástrojů pro český trh a přispěli k efektivnějšímu vedení daňové evidence pro malé podnikatele.

Klíčová slova: Bootstrap, daňová evidence, databátová aplikace, formulářová aplikace, PHP, programování, účetnictví, webová aplikace

Title:

Web Application for Registering Finances of Small Businesses

Author: Martin Čepela

Abstract: This bachelor's project deals with the complexity of tax records and the challenges that small entrepreneurs have to face. The goal of this bachelor's project was to create a clear website for keeping tax records of a small entrepreneur. The application is processed based on the corresponding laws for tax records of freely accessible accounting software applications. The appearance of the application was solved using the JavaScript open-source library Bootstrap. With this project, we responded to the absence of clear, specialized tools for the Czech market and contributed to more effective tax records for small businesses.

Key words: accounting, Bootstrap, database application, form application, PHP, programming, tax records, web application

Contents

Introduction

This bachelor project focuses on the topic of creating an application that would enable keeping tax records for small businesses in the Czech Republic. The market for accounting applications in the Czech Republic is very diverse. Most applications strictly focus on accounting, they usually do not have the option of keeping tax records and might be expensive for small businesses.

The aspects mentioned earlier suggest that the application should be unambiguous and easily accessible. Besides being free, the application should be also available to every user regarding their operating system or device. This will be achieved by publishing the application on the Internet. It should be considered whether the user's browser is compatible with the application. The solution should include responsiveness on mobile devices.

Accounting applications usually contain buttons, forms, alerts, pop-up windows, and menus. For this reason, there are several open-source libraries for creating such applications. One of them is called *Bootstrap* and it contains previously mentioned functions. With this library the application will look more organized, which fulfills initial assumptions. The JavaScript also provides dynamic behavior in tables, such as disabling buttons or sorting columns. The graph can be generated with the knowledge of *JSON*, which has predefined object notation given by the *Plotly* library. With the combination of these programming languages, a visually appealing user interface is assured.

The first chapter focuses on the theoretical part of the problem, which includes learning the fundamentals of accounting, flat-rate tax and tax records. The second chapter includes information about relevant applications, analyses the core functionalities that should appear in this application, and mentions the qualities that should appear in the application. The third chapter lists the technologies used to create this application and their definitions. The fourth chapter describes the class structure, the database structure, the user interface of the application and the application logic with a database and a structure scheme. The final chapter is focused on testing the application by selected persons.

Chapter 1

Recording Taxes in a Business

For an entrepreneur in the Czech Republic, there are three ways of registering taxes: accounting, tax records, and flat rate tax. The difference is determined by the amount of financial return in the previous tax period. Accounting is suitable for companies with income per the previous tax period over 25 million CZK, companies using the method of tax records cannot exceed the limit of 25 million CZK, and flat rate tax is intended for small businesses with income lower than 2 million CZK.[7]

1.1 Accounting

Obligatory method of keeping records of financial data in companies with income being greater than 25 million CZK per previous tax period. Accounting is understood as units that keep full-scale or simplified accounting accounts for the state and movement of assets and other assets, liabilities, including debts and other liabilities, as well as costs and revenues and the result of the economy with double entries. Accounting is done within the accounting period, or if necessary, in a different accounting period. The accounting period is understood as 12 consecutive months of the year starting from January. If the period is shorter than 12 months or does not start in January, it is no longer called an accounting period but a financial year. The accounting or financial period may be prolonged if the entity is established or extinct 3 months before or after the period. The responsible person for keeping financial statements and the accounting of trust funds is the manager, legal entity, or a person elected by the manager given in § 6, 563/1991.[2]

1.2 Tax records

In the Czech Republic, tax records are a way of registering income or expenses for companies whose income did not exceed the limit of 25 million in the previous tax period. They are composed of incomes and expenses, demands and debts, and fixed and minor assets. By recording this data the tax base is calculated.[5] Moreover, the way of keeping records is not important – by law, there is no explicitly said how to manage the record of data, only what data to keep. With the tax records, it is important to record any financial flow in or from company finances unlike in accounting where the date of the received or issued invoice is important.

In addition, the difference between accounting and tax records is that only an individual can keep tax records.[3]

1.2.1 Incomes and expenses

Income in a company is understood as a positive financial value increase of the business, such as financial revenue for a service provided by the company or providing a product, etc. The expense is the exact opposite – financial value negatively affecting the value of a company, e.g. equipment purchased for use in the company (losing value over time if tangible), investment, or software. The application should provide records of the following: the name of the transaction, document number, date of acquisition, information whether it is revenue or expense, and financial value. Additionally, it should also be evident, if the tax is included or not, if the transaction has been done from the bank or in cash, and a description for easier orientation.[7]

1.2.2 Demand and debt

Besides income and expenses, the business has to keep records of its demands and debts. Demand could be understood as an issued invoice for a provided service while debt is the opposite – an invoice issued by a different business to be paid. It is important to record the document number, the company's name, the date of acquisition, whether the object is demand or debt, financial value, the tax included, and the description.[7]

1.2.3 Property of a company

Fixed assets are any tangible or non-tangible subjects that have been purchased by the business and are depreciated over some time with their purchase price greater than 80,000 CZK. The purpose of keeping records of assets is to match the revenues of the business with its expenses over the lifespan of the asset, i.e. the asset is generating value for the business while it is being depreciated.

However, sometimes the asset fails or breaks, and therefore they cannot produce a value in their lifetime anymore. In this case, they have to be eliminated and the date of elimination is registered. The sum of depreciated values until this point is called accumulated depreciation.

For the application, the data to be stored is the item's document number, name, value, date of acquisition, date of withdrawal (could be blank), accumulated or straight-line method, a group to which the asset belongs, and a description.[8]

Fixed assets are divided into 6 fundamental groups by their value:

Depreciation group	Example	Number of years
1	Computers, tools	3
2	Cars, buses, most tools	5
3	Steam boilers, trams	10
4	Buildings, towers, pipelines	20
5	Industrial buildings, flats, agriculture	30
6	Administrative buildings, department stores, hotels	50

Table 1.1: Depreciation groups.[9]

1.2.4 Minor assets

Minor assets are properties purchased by the company with a value lower than 80,000 CZK and an estimated lifespan greater than 1 year. Minor assets are divided into tangible and intangible assets. Data that is to be recorded are the date of acquisition, the document number, and the date of withdrawal.[10]

1.2.5 Depreciation methods

The price of the property is gradually subtracted from the initial price. In the Czech Republic, there are two fundamental depreciation methods: Straight Line Depreciation and Accelerated Depreciation Method. The business owner can choose between these but normally the Straight Line Method is used for shorter lifespan assets such as computers, and the Accelerated Method for longer lasting assets such as buildings.[9]

The two methods are defined as:

Straight-line depreciation method

1. year:

$$A = \frac{(I \cdot F[j])}{100}\,, \tag{1.1}$$

kth year:

$$B = \frac{(I \cdot C[j])}{100}\,, \tag{1.2}$$

where:

I = initial price
j = depreciation group
F = percentage given by j (j=1)
C = percentage given by j ($j = k; k > 1$)

Group	Example	Years	F[%]	C[%]
1	Computers, tools	3	20	40
2	Cars, buses, most tools	5	11	22.25
3	Steam boilers, trams	10	5.5	10.5
4	Buildings, towers, pipelines	20	2.15	5.15
5	Industrial buildings, flats, agriculture	30	1.4	3.4
6	Administrative buildings, department stores, hotels	50	1.02	2.02

Table 1.2: Depreciation table with values for Straight Line Method.[9]

Accelerated method

1. year:

$$X = \frac{I}{f[j]}, \tag{1.3}$$

$$R = I - X - \sum_{n=2}^{k} Y, \tag{1.4}$$

kth year:

$$Y = \frac{2 \cdot R}{c[j]}, \tag{1.5}$$

where:

R = remaining value
I = initial price
f = first year coefficient
c = consequent year coefficient
j = depreciation group
X = first year
Y = consequent years

Group	Example	Years	f	c
1	Computers, tools	3	3	4
2	Cars, buses, most tools	5	5	6
3	Steam boiler, trams	10	10	11
4	Buildings, towers, pipelines	20	20	21
5	Industrial buildings, flats, agriculture	30	30	31
6	Administrative buildings, department stores, hotels	50	50	51

Table 1.3: Depreciation table with values for Accelerated Method.[9]

1.2.6 Accumulated depreciation

The term Accumulated depreciation is understood as a period during which a physical object has been registered in business and has been losing its value over the years until the year of withdrawal. The year of withdrawal can be determined by either table 1.1 where each object dependent on their value belongs to one group and is later depreciated over this period or the object is damaged or is unusable for the company. In this case, depreciation over the years for this object has to be summed up until the earlier date of withdrawal. The withdrawal date serves to determine the correct tax base for the individual at the end of the tax period.[11]

1.2.7 Money journal

A money journal calculates the tax base of a given period and keeps track of all the transactions. Particularly important items that should be displayed in the money journal are the date, name, document number, incomes, and expenses. The incomes and expenses are further divided into taxable and non-taxable. This division is essential to determine the tax base (sum of the incomes minus expenses in the given period).[5]

Other types of journals in tax records are Cashbooks, Bankbooks and register journals. To recognize the difference to which journal each income or expense belongs, income or expense type has to be recognized, i.e. personal deposit or withdrawal is saved in the money journal, cash transfer from a cash register to a bank will be in the bank journal and opposite of the previous will be saved into a registered journal.[12]

1.3 Flat-rate tax

This method for recording taxes in the Czech Republic was created in 2021 and is suitable for entrepreneurs with yearly incomes lower than 2 million CZK. This method is intuitive for individuals who do not want to keep tax records. Instead, their expenses are applied as a percentage. The conditions to be able to apply flat-rate tax given by the law are to be self-employed, not be a VAT payer, they are not a partner or debtor in the company, and the individual's income in the previous tax period did not exceed 2 million CZK.[6]

Additionally, the individual is obliged to file a notice to the trade office by filling form released by the Ministry of Finance. Moreover, according to Act No. 540/2020 Coll., in § 38lc the entrepreneur is required to keep records of his:

1. identification and contact details,
2. data on entry into the flat-rate scheme,
3. data on the taxpayer's pension insurance,
4. data on the taxpayer's public health insurance.

Nevertheless, if the entrepreneur is inclined to withdraw from the flat-rate regime, 2 elements have to comply. By Act No. 540/2020 Coll., in § 38le individual can resign by giving notice to the trade office before the 10th day of the upcoming tax period or stating in which tax period the individual would not be using a flat-rate tax regime anymore.[6]

Involuntary abdication occurs by Act No. 540/2020 Coll., in § 38lf if the individual is aware of not accomplishing the flat-rate tax requirements and must inform the tax administrator before the 15th day of this finding. If the individual is not aware of not fulfilling minimal requirements, the tax administrator is obliged to record an official record and to apprise the individual of this fact including reasons.[6]

Chapter 2

Analysis

This chapter evaluates the design, functional features, and security features of the application. First, four accounting-related applications are tested and analyzed. Then, a detailed procedure of necessary functions is listed. Finally, important security issues are named and described.

2.1 Market analysis

To have some overview of software in this field, a detailed analysis had to be done. In this chapter, the most useful and similar web or desktop applications related to accounting and tax records are listed and compared. Each application is evaluated based on design, the possibility of keeping tax records, and price.

2.1.1 POHODA

POHODA is a desktop/web application for managing accounting and tax records. It is easy to understand. It has a variety of versions and subcategories differing in accessibility and price: Mini, Lite, Jazz, Standard, Profi, Premium, and Komplet. Those versions, excluding Mini and Lite are subdivided into POHODA, POHODA SQL, and POHODA E1. However, categories Mini, Lite, Jazz, Profi, or Premium do not support keeping tax records and versions Standard — POHODA and Komplet. POHODA does not include the connection between the server and the client (SQL database), which is necessary for remote connection to tax records for the web application. Therefore, suitable versions would be Standard — POHODA SQL, POHODA E1 or Komplet — POHODA SQL, POHODA E1. Furthermore, the application might be very expensive – the cheapest variant of the suitable software for basic tax record keeping is 16,980 CZK and the most expensive with all of the functions is 34,980 CZK.[13]

After a consultation with a person using POHODA daily, known as Person 4 in this project, the following can be concluded:

- The system displays all the data on a single page making it easy to understand
- The application allows data transfer to be utilized in accounting (document types or document numbers)
- The application requires a complete makeover to improve its visual design and styling
- The application might be difficult to navigate for a beginner

In conclusion, POHODA is a universal desktop/web application designed for managing accounting and tax records. It offers a range of versions and subcategories, each customized to different user needs and budgets. However, it's important to note that not all versions meet the requirements for tax record keeping and some lack essential features including remote connectivity.

2.1.2 iDoklad

iDoklad is an online application with an easy-to-understand user interface. Initial login into the application offers help with creating the first user's invoice by showing pop-up windows over the screen. On the first page of the application, there is an easily understandable bar chart graph with months on the x-axis and values of invoices on the y-axis. Each month is further divided into two separate bars in the same month, the first one displaying the value of the successful sale and the second one the value of purchase. Furthermore, the graph enables displaying data in quarters and years, with or without VAT.[15] The homepage displays the balance per period and VAT registration. A small business license is 170 CZK a month, making it accessible in the short term.[14]

In summary, the application is intuitive for tracking invoices, it is an inspiration for styling the user interface, and it is relatively cheap. However, it does not contain functions for tax records.

2.1.3 Účtovna

Účtovna is a free web-based application that provides a platform for managing tax records. The website is very easy to navigate with non-redundant information on the screen. Moreover, it contains an easily understandable menu at the top of the site divided into invoices, documents, and three journals (money, cash, and bank), based on the beginning and end of the period, situated in a drop-down menu. Furthermore, there is a possibility of editing company information or user information.

After logging into the application for the first time, the homepage displays buttons in the body of the page for easy creation of new items. Normally, they are hidden in the top menu which makes the navigation easier for beginners. Moreover, after adding data to the application and returning to the homepage, buttons are still displayed on the page, two new pie charts appear and the individual last added items are displayed on the right side of the home page. The chart on the left provides information about income and expenses throughout the year while the right one displays a summary of income and expenses per all time periods.

However, there might be a few disadvantages in the navigation:

- Locating fixed and minor assets in the application might be challenging
- Each button shares the same color
- A chart for total income and expenses is absent

On the other hand, the application is capable of generating a preview of documents hidden under the document number reference link. In the document preview, the site displays individual actions: change the document, print the document, and cancel the document. [16]

In conclusion, the application is very user-friendly when it comes to keeping tax records and perhaps the best on the Internet for the Czech Republic. It provides all the necessary tax records functions while maintaining a simple user interface.

2.1.4 iÚčto

iÚčto serves a similar purpose to iDoklad, allowing users to store essential accounting information. However, despite the same functions as iDoklad, its design looks less user-friendly by displaying multiple graphs on the front page, with detailed information under them. The application is also similar to iDoklad in the possibility of annual or monthly payments at around 300 CZK per month for the Standard version including the free trial version.[17]

In conclusion, iÚčto has several functions to record accounting in detail while maintaining a little less user-friendly interface compared to iDoklad.

2.1.5 Conclusion

In summary, the most intuitive application for keeping tax records in the Czech Republic is Účtovna which contains all the necessary functions for keeping tax records. Furthermore, it is for free and has a simple user interface. It is dedicated to the payers of VAT.

POHODA, unlike Účtovna, has the option of bookkeeping. However, the application is more expensive and the user interface is outdated. iDoklad and iÚčto are not sufficient for keeping tax records.

2.2 System features

2.2.1 Functional requirements

Functional requirements define software functionality. This application should be able to do the following:

- Allow user to login
- Allow user registration
- Edit user profile, change password
- Show a graph on the homepage
- Summarize income and expense for this year
- Show a total difference between income and expense
- Keep necessary data related to income and expense
- Keep data of receivables and payables
- Keep a record of minor and fixed assets
- Allow editing and deleting data
- Show data given in the chosen period in the journal
- Allow table sorting
- Calculate the tax base
- Depreciate fixed assets
- Display a help page

- Calculate the elimination of the asset
- Allow the user, when the asset is sold, to input a price and date of the sale
- Save each compensated demand or debt as income or expense

2.2.2 Non-functional requirements

Non-functional requirements are defined as the properties and qualities of the system. They do not include the functionalities of the system and determine what should be possible without user input. The non-functional requirements are as follows.

- Visibility on the Internet
- Responsible and modern User Interface
- Easily navigable User Interface
- Support of all internet browsers
- Security of the website
- Present the data within a reasonable timeframe
- Code extendibility

2.3 Use case

First, we introduce the use cases and related concepts, and then we list each of the individual use cases of the application that is to be developed. The first appearance of the term for buttons, for example, will be supplemented with the Czech name as it is displayed in the UI.

2.3.1 What is a use case

Use cases describe the ways the application will be used. In the description of use cases, we use the following terms:

- *Actor* is anything or anyone who triggers the function of the application.
- *Preconditions* are conditions that have to be accomplished before the sequence of the use case.
- *Triggers* are events that initialize the sequence.
- *Sequence* are events that happen in the application after meeting the preconditions and if the use case's triggers align, everything proceeds smoothly.
- *Alternative* flow is a flow from the point in the sequence, where some behavior of the application is not aiming to fulfill post conditions of the use case.

Each of the use cases will print an error message if they are not accomplished. Alternative flows may begin with a higher number than 1. This means that points with lower numbers are repeated.

Creating a user account

- Actors: user, system
- Preconditions: user is on the *Login* page
- Triggers: user clicks on a *Create a new account* ("Vytvořit nový účet") hyperlink
- Flow:
  1. User fills text in fields username, email address, password and confirms password
  2. User clicks on the *Signup* ("Registrace") button
  3. System checks if all the fields are filled with data
  4. System checks data with regular expressions
  5. System saves data to the database
  6. System redirects the user to the login page
- Alternative Flow
  2. System finds out one or more of the filled fields contain wrong data
  3. System outputs an error message:
     - Username already exists
     - Wrong format of the mail
     - Passwords are not matching
     - Password has to be longer than 8 characters
     - Password must contain at least 1 letter or number

Log in to the website

- Actors: user, system
- Preconditions: user is on the *Login* page
- Triggers: user clicks on the *Login* button or a *Register* page, or opens the website
- Flow:
  1. User fills username and password fields
  2. User clicks the *Login* button
  3. User checks if all fields are filled with data
  4. System checks if the input username matches any username in the database and at the same time if the password belongs to the username's ID
  5. System saves data to the database
  6. System redirects the user to the homepage
- Alternative Flow
  2. System recognizes wrong input data
  3. System prints an error message

Displaying income and expense difference throughout the year

- Actors: user, system
- Preconditions: user is on the homepage, the database contains previously added data by the user
- Triggers: user selects a year
- Flow:
  1. System displays a list of years
  2. User selects a year
  3. System selects all data for this year
  4. System calculates and displays the sum of the values

Adding income or expense and claims or debts

- Actors: user, system
- Preconditions: user is logged in and is on the page with income and expense or claims and debts
- Triggers: user clicks on the *Add* button
- Flow:
  1. System shows a pop-up window
  2. User fills in the necessary data
  3. User clicks on the *Save* button
  4. System checks for missing and incorrect data
  5. System saves the correct data into the database
  6. System closes the pop-up window
  7. System displays a table with the added item
- Alternative Flow 1
  2. User sets the demand or debt as "paid"
  3. System saves the demand or debt as income or expense

  Note: minor assets are added as an expense

Adding fixed assets

- Actors: user, system
- Preconditions: user is logged in and is on the page with income and expense, claims and debts, fixed assets or minor assets
- Triggers: user clicks on the *Add* button
- Flow:

1. System shows a pop-up window
2. User fills in the necessary data and marks the asset as tangible with a price greater than 80,000 CZK or user sets the value of the asset lower or greater than 80,000 CZK but the asset is non-tangible
3. User clicks on the *Save* button
4. System checks for missing and incorrect data
5. System saves the correct data into the database
6. System closes the pop-up window
7. System displays a table with the added item
8. System saves the asset as an expense

- Alternative Flow 1

2. User sets the value of the asset lower than 80,000 CZK
3. System disables the save buttons

Displaying tax base in the journal

- Actors: user, system
- Preconditions: user is on a journal page
- Triggers: user clicks the *Submit* button
- Flow:

1. User selects the start and end of the time period
2. User clicks on the *Submit* button
3. System generates a table with data
4. System displays the sum of the tax base

Editing income or expense, claims or debts, fixed assets and minor assets

- Actors: user, system
- Preconditions: user is on one of the records page
- Triggers: user clicks on the *Edit* button
- Flow:

1. System loads a pop-up window
2. System loads the information of the table row
3. System disables the fields with the depreciation group and the depreciation method
4. User does or does not make changes
5. User clicks on the *Save* button
6. System updates the database data

7. System closes the pop-up window
8. System refreshes the site

- Alternative Flow

  2. User updates the value of the minor asset to be higher than 80,000 CZK
  3. System enables the fields with the depreciation group and the depreciation method

Deleting income or expense, claims or debts, fixed assets and minor assets

- Actors: user, system
- Preconditions: user is on one of the records page
- Triggers: user clicks on the *Delete* button
- Flow:

  1. System loads a pop-up window
  2. User clicks on the *Submit* button
  3. System deletes data from the database
  4. System closes the pop-up window
  5. System refreshes the site

Depreciation of fixed assets

- Actors: user, system
- Preconditions: there is an existing fixed asset older than one year
- Triggers: user is logged in
- Flow:

  1. System goes through the database and compares fixed asset dates
  2. System marks depreciated assets as an expense and inserts it into the database with the value dependent on the depreciation group and depreciation method

Elimination of fixed assets

- Actors: user, system
- Preconditions: there is an existing non-eliminated fixed asset
- Triggers: user clicks on the *Eliminate* ("Vyřadit") drop-down button
- Flow:

  1. System unwraps 2 options: *Disposal* ("Likvidace") or *Sale* ("Prodej")
  2. User selects *"Disposal*

3. System stops the calculation of the upcoming depreciation and sets the depreciation value for the current year as the remaining value, the remaining value is set to 0

- Alternative Flow

2. User selects *Sale*
3. System opens a pop-up window with the date and value of the sale
4. User fills in the data and clicks on the *Save* button
5. System stops the calculation of the upcoming depreciation for this asset and sets the depreciation value for the current year as the remaining value, the remaining value is set to 0
6. System adds the sale as an income

Log out

- Actors: user, system
- Preconditions: user is logged in the application
- Triggers: user clicks on the *Logout* button
- Flow:

1. System destroys the user session
2. System redirects the user to the login page

2.4 Quality attributes

This section is focused on the back-end features that are required to be robust for such an application. This includes the application's security and modularity. The individual subchapters were selected because I had encountered them whíle developing the application.

2.4.1 Security

A website related to tax records is prone to have several security vulnerabilities. The first includes a secure transmission of the data on the internet. This will be handled by *HTTPS* protocol available for purchase from a web hosting provider. Moreover, the data on the website should strictly be sent with a *POST* protocol which will eliminate unnecessary exposure of secret data in the URL to anyone using the application. The data should be sent with PHP's functions *prepare and execute*, securing safe data passage to the database which will avoid SQL injections.[18] Furthermore, the user passwords should be hashed. *MD5* algorithm will prove useful in this case because there is already an available function in PHP for this algorithm. For additional details on security, please refer to the 3.2 section.

2.4.2 Modularity

The code of the application should be legible and easily understandable. In this case, a possible solution is to structure the code in classes. This will make the code easily extensible, reduce redundant code, and ensure the safety of encapsulated data. The classes of the application are listed in the 4.1.1 with a scheme.

Chapter 3

Technologies and Security

The chapter lists technologies that I have used to create and manage this application. They are the following: HTML, CSS, PHP, JavaScript, JSON, jQuery library, Plotly library, AJAX, Bootstrap library, SQL, GitHub, phpMyAdmin, FTP, and XAMPP.

3.1 Technologies

HTML and CSS

Hypertext Markup Language is a standard markup language for creating websites on the World Wide Web to be published on the Internet.[19] The final version available is HTML5 (2017) with the last update 5.3. released in 2021.[20]

The styling of the HTML code can be enhanced using Cascading Style Sheets (CSS). CSS is a language designed to control the layout and styling of HTML elements. It was created due to the limitations of HTML styling standards. The latest version is CSS3 and is compatible with HTML5.[22]

In summary, HTML defines the structure of the website with individual elements and CSS is used to style them.

Bootstrap

Bootstrap is a HTML, CSS and JavaScript API that enhances the style of HTML such as forms, buttons, or icons.[23] The main role of Bootstrap in this application is to facilitate the creation of the appearance of various inputs without complex and long CSS classes.[24] To make Bootstrap available in the application, the included link is to be added in the head element of the HTML document or at the top of each page.

Moreover, Bootstrap's elements can be further edited with Sass scripting language integrated into the CSS, however, in this application, Sass was not necessary.[24] The bootstrap version for this application is 4.3.1.

PHP

PHP is a scripting language, that offers key functions to connect to the database. PHP is mostly used to create internet applications such as Wikipedia or Facebook. It is one of the most used programming languages in the world for web development due to its extensive list of functions.[25] The latest version of PHP is 8.3.[25] It provides the structure of the code, algorithms, and access to the database. The application is developed in PHP 8.1. as well as the web hosting server.

JavaScript

JavaScript is a scripting language useful for implementing dynamic websites. This includes displaying various behaviors of the site in real time without requiring a page refresh.[26] In this application, this could include functionalities such as sending data to pop-up windows, disabling or enabling buttons based on the user input, or displaying the graph which is adjusted to the user input. It is similar to C programming languages; the last version was released in 2018.[26] This project uses 2 JavaScript libraries: jQuery and JSON.

jQuery is a JavaScript library published in 2001 and is widely used for creating website content such as pop-up windows, site animations or event handling.[27] JQuery can be added with HTML's <link> in the <head> tag. JQuery is also capable of performing AJAX requests.

JSON (JavaScript Object Notation) is a library known for its structured and extendable code and useful for retrieving data from the database. This makes it relevant for creating database-based graphs. The code is realized in pairs and arrays.[28]

AJAX

Asynchronous JavaScript and XML is a scripting language for dynamic web development. AJAX's main function is displaying data asynchronously which means only one task can be shown at a time and can not overlay any other.[29] In this application, it is responsible for sending data to pop-up window forms.

SQL

Structured Query Language is a programming language for creating databases. SQL is designed to interact with inserting, editing or deleting data from the databases. MySQL is a related open-source database model offering organized and clear work with the databases generated by SQL code. Moreover, SQL tables may be connected by primary and foreign keys.[30] The tables may be created by either the code or in the User Interface of phpMyAdmin where SQL can be managed.

PhpMyAdmin

Is a free software created in PHP for easy maintenance of SQL databases. For this project, phpMyAdmin serves various functions, including:

1. The ability to perform tasks such as creating, deleting, and editing database tables.
2. The option to interact with individual table rows, simplifying data modification.
3. The assistance in the management of primary and foreign keys.

[31]

PhpMyAdmin was utilized for both local and public testing of the web application, making it an integral part of the development process. During the beginning of the website development, the XAMPP open-source Apache distribution was used for the facilitation of database management on a local server.[32]

FTP

File Transfer Protocol is a protocol for sending data between a user and a server. Files can be either uploaded or downloaded from the server, making it easy to manage a website.[33] Multiple applications support FTP connection, such as Total Commander for Windows or Double Commander on Mac. FTP can be also integrated into the developer software. The difference is that the saved code can be instantly sent to the server, substantially reducing the time needed to transfer files.

In this case, to connect to the FTP server, Visual Studio Code extension *ftp-simple* was used. It is an easy-to-use tool that can be accessed by pressing "F1" in VS Code. The following steps were accomplished by following a tutorial.[34] First, *ftp-simple* can be included in the VS Code by searching and downloading in "Extensions". Then, it is necessary to set server attributes to the FTP config file. This can be done in "ftp-simple: Config" after pressing "F1", which opens a JSON file. There is a prefilled set of variables: *name, host, port, type, username, password, path, autosave, confirm*. The *name* variable can be set to anything the user prefers, the *host*, the *username* and the *password* are given by the webhosting provider, the *port* is set as a default to 21 and *type* to "FTP". Furthermore, The *path* serves as a backup file path that can be left empty. The *autosave* and the *confirm* option are booleans. On "true" the *autosave* sends data to the server immediately after the user saves the file and the *confirm* on "true" displays a pop-up window to confirm the save of the file.

Then, after the correct server configuration, press the "F1" button again to display the search menu and find "ftp-simple: Remote directory open to workspace". This displays the selection of the website by the name set in the JSON file and represents the FTP server directory. After navigating to the desired path, press the "Enter" button.

From now on, the connection is established and changes in the website's code can be directly made by either saving the files or creating new files. The changes will appear instantly after the page refresh.

GitHub

GitHub is a worldwide software for versioning and collaboration providing Git repositories. Each GitHub project contains a master branch to which data is committed and afterward pushed, creating a viewable comparison of code differences with a commentary. Another feature of GitHub is a pull request, allowing users to download the data from the server and synchronize it with local data. It enables retrieval of older files that could be shared by other members from the same branch if needed.[35]

In this work, I am using GitHub to access older versions of the application if needed. This always gives a backup option when the code stops working completely and backtracking is required. The application's code can be found under the following hyperlink: `https://github.com/Martyneqq/bc`.

3.2 Security

The following technologies are used in this application to ensure better security and protect vulnerable database data. These are all of the found security vulnerabilities during the application development.

HTTPS

The data transmission on the Internet between the client and the server is managed by HTTPS protocol. The client requests a server certificate and sends encrypted data.[36] The data is encrypted with the Diffie–Hellman key exchange method, developed in 1976, instead of sharing public keys in HTTP which can be exposed to a third party. The code combines the private variables of each side with public variables, creating its key that can be decrypted by both sides without sharing data with a third party. The Diffie-Hellman principle is based on a discrete logarithm problem in discrete mathematics, which is notable for its challenging solution complexity when reversed.[37]

- Let G be a finite group generating a number g.

- Let a and b be random private numbers on each side.
- First side shares its private number in this equation: $A = g^a$.
- Second side shares $B = g^b$.
- After A and B are shared as public numbers, the first side does equation $X_1 \equiv B^a$, and the second side does $X_2 \equiv A^b$.
- The numbers X_1 and X_2 are equal, efficiently creating a public encrypted variable, solvable by each side. The third side can not solve the equation since it does not have access to a and b.

[38][39]

In the internet application, without HTTPS security available, the page will display a pop-up window that the website is insecure, discouraging users from using the website to avoid getting important data stolen.

POST

In HTTP, there are 2 fundamental request methods to send data to the server: GET and POST. Both are accessible in PHP. The difference between them is in the way of updating server data since GET has its variables visible in the website's URL, including private data such as passwords, names, or numbers. This makes the GET method useful for passing information about data unrelated to the server such as page numbers.

POST, on the other hand, pushes data to the server in the body of the code, not the URL, bypassing GET's disadvantage.[40]

MySql prepared statements

When data in a database is mishandled, it becomes prone to "injection" attacks, where the user can exploit the code by passing an apostrophe character in the site's input forms, allowing SQL code to be closed and executing unauthorized code afterward.

In PHP, variables can be sent to the server and used in SQL queries. However, to prevent SQL injection vulnerabilities, it is recommended to use prepared statements and parameter binding. With prepared statements, variables are bound as parameters to the query, which ensures that the user input is properly sanitized, and prevents malicious SQL injection attacks.[41]

MD5

Message-Digest algorithm is a function known for hashing input data to a fixed length. The algorithm is designed in a way minor adjustments to the input result in a significantly different output, making the data irreversible. It is usually used to encrypt user passwords.[42] However, a significant amount of data encrypted with MD5 has already been decrypted and is accessible online.

The sequence to ensure the correct password:

- User inserts the password
- MD5 hash encrypts the password
- System sends the encrypted password to the server
- After repeated Login, MD5 hash is applied to the inserted password once again

- Sent password to the server is compared with the one saved in the database.
- If they match, the passwords are correct

Chapter 4

Application Architecture

4.1 Structure

This chapter is focused on the structure of the application. The code is structured in classes, which makes the code more legible, separates the database access from the application logic and reduces the code repetitiveness. Classes are ordered alphabetically.

First, the class diagram is shown, which is followed by a detailed description of each method in the desired class. The first appearance of the term for buttons, for example, will be supplemented with the Czech name as it is displayed in the UI.

4.1.1 Class diagram

This graph visualizes classes and their relations. There are two types of connections in the graph: with and without arrows. The lines with arrows signify an inheritance, where the class that is pointed at is the one inherited. The lines without arrows stand for has-a relation, where one of the classes uses the methods of the other. The class diagram was generated by *Gleek.io*.

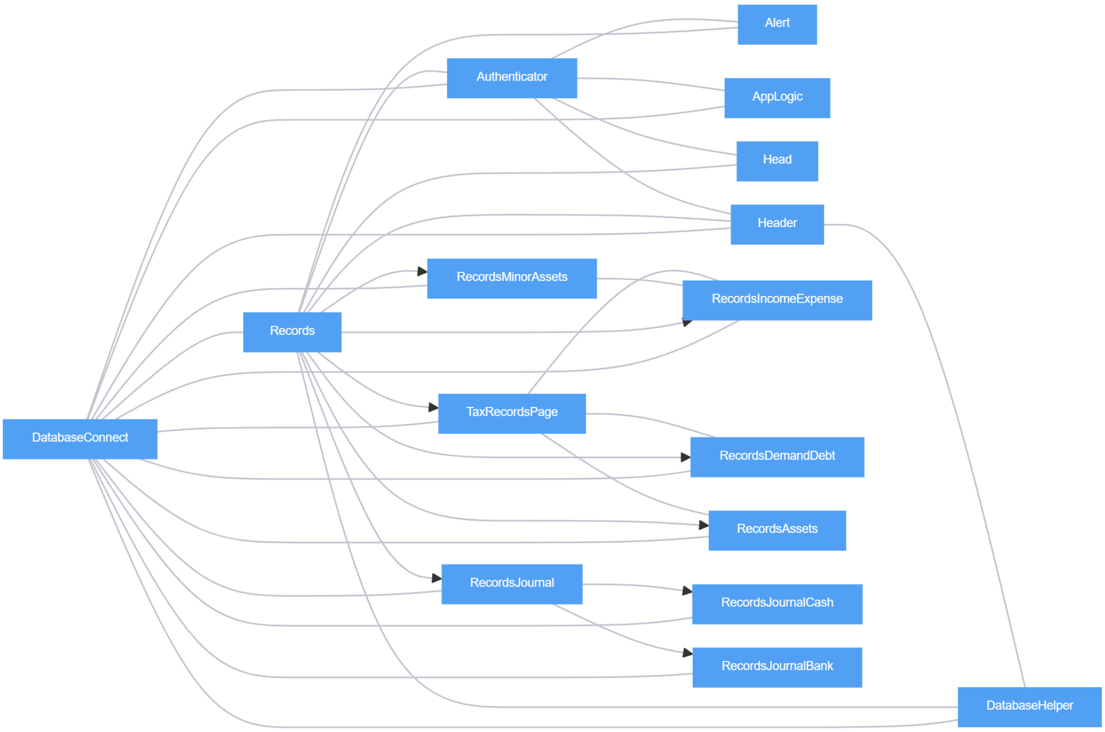


Figure 4.1: Classes of the application.

4.1.2 Classes

This chapter describes the methods and functionality of each class.

DatabaseConnect

The core script, which contains information about the server name, username, server password, and database name, stored in the *connect* variable. This variable is essential in later connections for all classes using the database connection. This script is included in every class excluding *Head* and *Alert*.

Authenticator

Contains methods *Check, Login, Signup, Logout* and methods for generating HTML-related code for signup and login pages. This class is intended for storing login-related methods.

- *Check* – keeps the user login on the website, which means that the user session is verified each time the page is changed.
- *Signup* – creates a user after they insert their *identification number, username, e-mail and password* into the forms on the signup page. If any of the input data does not satisfy regular expressions, it sets a session with an error message.

- *Login* – looks for the *username* that matches the *password*. If the *username* or *password* are inserted incorrectly, it sets an error session the same way as in the *Signup* method. This session is then used in the *Alert* class.

Alert

Generates an alert related to an action. These actions can be adding an item, editing an item, depreciating the item, deleting an item, or eliminating the item. This class contains only two methods: *GenerateAlert* and *Alert*.

- *GenerateAlert* – creates a dynamic HTML message.
- *Alert* – updates the style and displays the correct message according to the session that has been triggered.

This class has a *has-a* relationship with the following classes: *Authenticator, DatabaseHelper* and all *Records* classes, which means, only these are able to generate an alert.

DatabaseHelper

Generally used for the database access. This class has a *has-a* relationship with every class excluding the following classes: *Journal, Head, Alert*. It contains the following methods:

- *TotalYearlyValue* – retrieves a list of years related to the income or expense in the database. This method is used in the *TaxRecordsPage*.
- *YearlyIncomeExpense* – loads the total value of income and expense depending on the value of the *TotalYearlyValue* method. This method is used in the *TaxRecordsPage*.
- *TotalFlow* – displays the cash and bank balance used in the *Header* class.
- *YearlyDataForGraph* – loads all necessary data for the graph generation. This method is used in the *TaxRecordsPage*.
- *SubmitInsertData* – inserts forms filled by the user into the database. Receives *name* as an argument. The argument stands for the name of the *Save* ("Uložit") button. The method of the *DatabaseHelper* class switches between different buttons to save the desired item from the page to the database. This method is used in the *TaxRecordsPage* and the classes extending *Records*.
- *GetTableRow* – gets the id of the table row where the *Edit* ("Upravit") button is located. Then displays the data from the database in each form. The code length is reduced by passing the table name as an argument. This method is used in all classes extending *Records* (is-a relationship).
- *SubmitEdit* – updates the data edited by the user by clicking the *Save* button. This method is used in all classes extending *Records*.
- *SubmitDelete* – deletes selected table row by clicking the *Delete* ("Smazat") button. This method is used in all classes extending *Records* (is-a relationship).

Head

Contains meta tags, links, scripts and the title of the page, which can be normally found in the <head> meta tags in the HTML. It is included in the following classes: *Authenticator* and every class extending *Records*. In summary, this class is used only where the HTML content is rendered.

Header

Displays the top menu on the website. It consists of the following methods:

- *CashFlow* – gets the total value in cash from the *DatabaseHelper's* method *TotalFlow*.
- *BankFlow* – gets the total bank balance value from the *DatabaseHelper's* method *TotalFlow*.
- *Guide* – displays the manual.
- *RenderHeader* – displays the top menu (including the three previous methods).

It is used in the same classes as the *Head*. Moreover, this class uses the *TotalFlow* method from the *DatabaseHelper*.

Records

Parent class of the *RecordsIncomeExpense, RecordsDemandDebt, RecordsAssets, RecordsJournal, TaxRecordsPage*. The methods are the following:

- *GetHead* and *GetHeader* – used for the retrieval of the head and header variables.
- *GenerateDocument* – used for the generation of the document numbers. It ensures the correct mapping of the document names based on the user input.
- *Secure* – ensures that no malicious data can be inserted into the input forms, such as SQL injection.
- *GetColor* – changes the color of the value in the journal to green or red depending on whether the value is income or expense.

Furthermore, the class has instances of objects *Head, Header, Alert* and *DatabaseHelper* internally created in the constructor. Methods of these classes are later used in the inherited ones. An instance of this class is not created.

RecordsIncomeExpense

A directly inherited class from the *Records* containing the following methods:

- *TableHead* – dynamically creates a table head dependent on the number of columns. Uses a script *sort.js* to allow column sorting.
- *TableBody* – generates the body of the table by accessing the database.
- *TableFooter* – generates a sum of the values at the fixed bottom of the table.
- *RenderTitle* – creates a title above the table with the *pageTitle* variable, which is set in the class's constructor and directly set from the object.
- *RenderButtonAdd* – generates an *Add* ("Přidat") button above the table.
- *ModalBodyAdd* – displays the collection of forms in the pop-up window.
- *AddIncomeExpense* – creates a pop-up window with the *ModalBodyAdd* function.

- *EditIncomeExpense* – creates a body of the page which contains the forms for editing the item and accesses the database with the *DatabaseHelper's* method *GetTableRow* to load the data into the form fields.
- *RenderBodyContent* – this method is the body of the page. First, the content is stored in the *universal-table* <div> which is the same for the *Records* inherited classes. This makes it easy to structure the style of the body. In the <div>, there is a method of the *Alert* class *Alert*, *RenderButtonAdd* and *RenderTitle*. Then the method has a <div> *table-container* which executes the *TableHead, TableBody* and *TableFooter* methods to display the table as a whole. Outside the table is the *AddIncomeExpense* method (the modal window) – this is because the pop-up window must always be at the bottom of the <body> tag.
- *Render* – determines the present page (*Edit* or *Records* page) and generates the site accordingly with *EditIncomeExpense* or *RenderBodyContent* methods.
- *RenderHTML* – generates basic HTML structure with <head>, <header> and <body>. In the <head> and <header> tags, it internally accesses the *RenderHead* and *RenderHeader* methods from the *Head* and *Header* classes. In the <body>, the *Render* method is used.

The same design is also used in the *RecordsDemandDebt, RecordsAssets* and in the *RecordsJournal*

RecordsDemandDebt

A class inherited from the *Records*. It includes the same methods as the *RecordsIncomeExpense* but the data content is different.

RecordsAssets

A class inherited from the *Records*, which contains the same methods as the *RecordsIncomeExpense* and one additional:

- *ShowDepreciation* – passes the data loaded in the tables row into the modal window that displays the asset depreciation with the AJAX script.

RecordsJournal

A class inherited from the *Records*. However, it does not have a has-a relation with the *DatabaseHelper* and *Alert* because it does not use any of their methods. This class is intended for generating the journal page. It only contains *TableHead, TableBody, TableFooter* and *RenderBodyContent* methods, which are structured the same way as in the previous 3 classes. The table here is generated after clicking on the *Submit* ("Potvrdit") button. This button executes the script located in the *TableBody* and selects the data from the database in the selected time period in each of the date fields (start, end).

TaxRecordsPage

An inherited class of the *Records* and is the home page of the website. Besides the *Head, Header* and *DatabaseHelper* classes, it has access to the *RecordsIncomeExpense, RecordsDemandDebt* and *RecordsAssets*. I have done this because of the presence of the "quick navigation's" buttons to allow access to the individual pop-up windows for adding the data, i.e. *SubmitInsertData* method.

The methods are the following:

- *RenderTitle* – displays the title
- *GetYearData* – uses the *DatabaseHelper's* method *TotalYearlyValue* to display the list of the years in the selection list.
- *RenderQuickNavigation* – generates quick navigation with the buttons *Add* and *Display* ("Zobrazit").
- *RenderYearlyTotal* – loads the total values from the database for the relevant year with the method *YearlyIncomeExpense* from the *DatabaseHelper*.
- *RenderGraph* – prepares the data for the graph with the method *YearlyDataForGraph* from the *DatabaseHelper* class and then renders the graph with the use of JSON.[45]
- *Render* – establishes the fundamental structure of the page with the "container-child" inside the "container-upper" or "container-lower" <div>, and "container-upper" and "container-lower" are put in the "container-parent" tag. Furthermore, the methods *SubmitInsertData* from the classes *RecordsIncomeExpense, RecordsDemandDebt* and *RecordsAssets* are executed here.
- *RenderHTML* – renders the final body of the page. This includes the head, header and the *Render* method.

Furthermore, the page is separated into 2 <div> tags: "container-upper" and "container-lower". Both are under the "container-parent" <div>, which is also used for the table structure for other classes.

AppLogic

A class containing an algorithm for dynamically generating the asset's depreciation. It contains the following methods:

- *ExecuteDepreciation* – a method containing the depreciation logic. It is explained in the chapter 4.1.5.
- *SaveDepreciation* – saves the depreciation into the database table.
- *CheckIfDepreciationThisYear* – looks for the occurrence of this year's date in the database where the asset has the same *id*.
- *IsLastYear* – looks for the boolean named *lastYear* in the database. It is equal to 1 if the value representing the number of rows is equal to the number of years for the specific asset. Otherwise, the value is 0.

In summary, this class ensures the correct calculation of the asset's depreciation and saves it into the database table with the help of *IsLastYear* and *CheckIfDepreciationThisYear*.

Additionally, in the "js" repository, there are 9 JavaScript files ensuring the dynamic functionality of functions in the application. They are the following:

1. darkMode – invert the colors of the website. Each element that should change color is saved in the session to avoid clicking the color change each time the page is changed. When the user clicks the button, colors switch.
2. getAssetIDSale – get id of the asset for the *Sale* ("Prodej") window in the elimination.

3. jquery-3.6.1.min.js – a downloaded jquery script for the functionality of pop-up windows, bootstrap, etc. Available at jQuery.com.[46]

4. restrictAddAsset – disable *Save* buttons when the fixed asset is inserted as tangible while the price is lower than 80.000 CZK.

5. restrictAddAssetMinor – disable the save button when the purchase price higher or equal to 80,000 CZK is inserted. Furthermore, the item is set to expense without an option to change.

6. restrictEditAsset – while editing the asset, disable all fields except name, description, and tax to avoid wrong future depreciation calculations.

7. showDepreciation – load ID and name of the asset into the window.

8. sort – bubble sort algorithm used for sorting table columns. This includes sorting document numbers, numbers, *dd-mm-YY* dates, and other strings. Original version available at W3Schools.[44]

9. updateRow – sends data about the asset to the *select_modal_info.php* script.

Finally, to display each page, the instance of the class is created in every individual *.php* script. These scripts have the same name as the URL of the website. Each class is in the file named "class".

4.1.3 Database

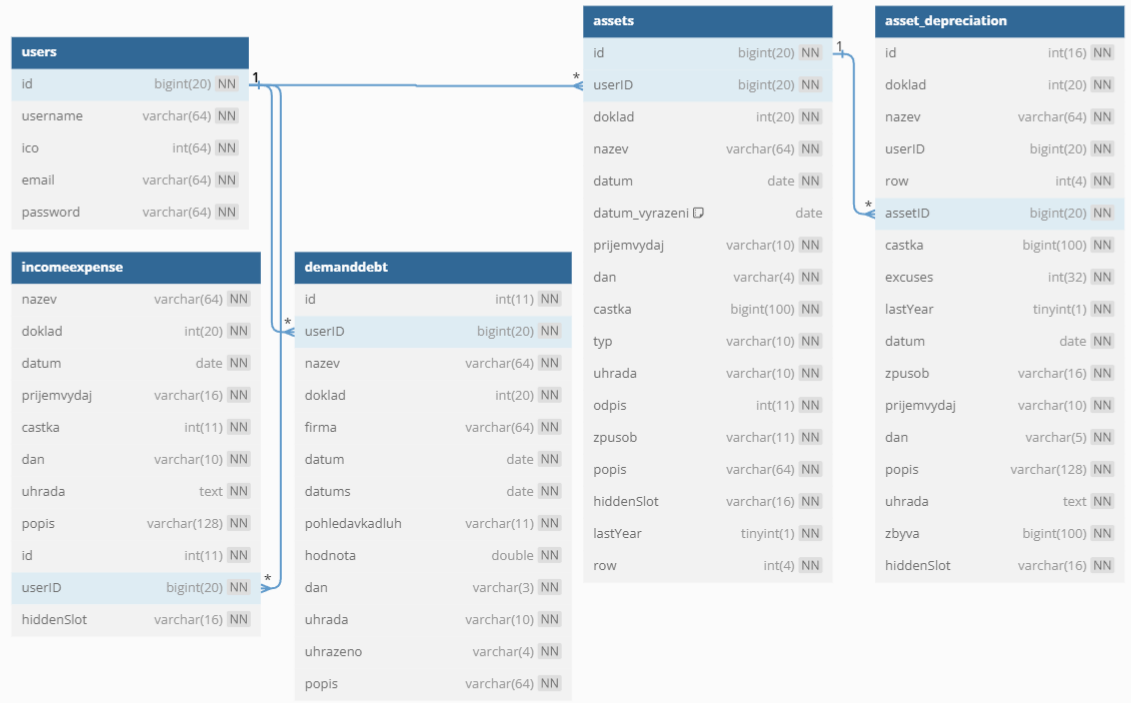
This chapter explains in detail how database connection is implemented in this application. In the scheme, tables *incomeexpense*, *demanddebt* and *assets* are connected to the *users* table by the id. This ensures that for each user, their data is displayed. The *incomeexpense* table is used to generate data for the *Income and Expense* ("Příjmy a výdaje") page. Moreover, the minor assets are saved in this table as an expense. The *demanddebt* is used to display data in the *Demand and Debt* ("Pohledávky a závazky") page. When the user updates the status of the *Compensation* ("Úhrada") column to *Yes* ("Ano"), it is also saved to the *incomeexpense* table. The fixed assets are automatically saved as an expense.

The table *assets* is used to display the individual assets in the *Fixed assets* ("Dlouhodobý majetek"). Individual depreciations in the pop-up window are retrieved from the *assetDepreciation* table, which needs *assets'* id to determine which asset data to show.

Furthermore, in *Income and Expenditure Journal* ("Deník příjmů a výdajů") there is a list of all incomes or expenses – fixed assets and a compensated demand or debt are counted as income or expense and are included here. The journal then lists all the data saved in the *incomeexpense* table for the selected time period. Some values contain *Depreciation/Elimination/Minor asset* ("Odpis/Vyřazení/Drobný majetek") before the *Item name* ("nazev") field. This was done to make the individual data more readable. The key variable here is *hiddenSlot*, which keeps the previous names to create the separation.

Finally, the *incomeexpense* table is used for many methods in the *DatabaseHelper* database table to generate the graph, outputs of the sum of the cash and bank value at the top, or calculates the sum of income and expense on the home page. The database scheme generated by `dbdiagram.io`.

The picture below shows the database scheme of this application. It includes information about table names, table relations, variable names, variable types and whether they are null or non-null values.



users		
id	bigint(20)	NN
username	varchar(64)	NN
ico	int(64)	NN
email	varchar(64)	NN
password	varchar(64)	NN

incomeexpense		
nazev	varchar(64)	NN
doklad	int(20)	NN
datum	date	NN
prijemvydaj	varchar(16)	NN
castka	int(11)	NN
dan	varchar(10)	NN
uhrada	text	NN
popis	varchar(128)	NN
id	int(11)	NN
userID	bigint(20)	NN
hiddenSlot	varchar(16)	NN

demanddebt		
id	int(11)	NN
userID	bigint(20)	NN
nazev	varchar(64)	NN
doklad	int(20)	NN
firma	varchar(64)	NN
datum	date	NN
datums	date	NN
pohledavkadluh	varchar(11)	NN
hodnota	double	NN
dan	varchar(3)	NN
uhrada	varchar(10)	NN
uhrazeno	varchar(4)	NN
popis	varchar(64)	NN

assets		
id	bigint(20)	NN
userID	bigint(20)	NN
doklad	int(20)	NN
nazev	varchar(64)	NN
datum	date	NN
datum_vyrazeni	date	
prijemvydaj	varchar(10)	NN
dan	varchar(4)	NN
castka	bigint(100)	NN
typ	varchar(10)	NN
uhrada	varchar(10)	NN
odpis	int(11)	NN
zpusob	varchar(11)	NN
popis	varchar(64)	NN
hiddenSlot	varchar(16)	NN
lastYear	tinyint(1)	NN
row	int(4)	NN

asset_depreciation		
id	int(16)	NN
doklad	int(20)	NN
nazev	varchar(64)	NN
userID	bigint(20)	NN
row	int(4)	NN
assetID	bigint(20)	NN
castka	bigint(100)	NN
excuses	int(32)	NN
lastYear	tinyint(1)	NN
datum	date	NN
zpusob	varchar(16)	NN
prijemvydaj	varchar(10)	NN
dan	varchar(5)	NN
popis	varchar(128)	NN
uhrada	text	NN
zbyva	bigint(100)	NN
hiddenSlot	varchar(16)	NN

Figure 4.2: The database scheme of the application.

4.1.4 User Interface

The user interface is divided into the following 7 groups. Each group contains the same styling defined in the CSS or the Bootstrap library.

Menu

The menu is created with the Bootstrap class *navbar*. This class makes the styling intuitive since there is a predefined structure that is well documented on the Internet. Each function would have to be otherwise defined in CSS and JavaScript manually.

The menu consists of the button with the house icon, the drop-down button *Documents* ("Doklady"), which then unwraps to *Income and Expense* and *Demand and Debt*, drop-down button *Assets* ("Majetek") unwrapping *Fixed* ("Dlouhodobý") and *Minor* ("Drobný"), and a drop-down button *Journal* ("Deník") displaying individual journals: *Income and Expenditure journal, Cash book, Bank book*. In the middle, there are two different values – the left one is for the cash flow and the right one is for the bank flow. On the right side, the drop-down button with the user name is located. After clicking the button, 4 items appear:

- *User profile* – not implemented yet. This function will be considered in the future work.
- *Guide* – a help window displaying the content from the *User guide* ("Uživatelská příručka"). The body is structured in the class from the Bootstrap called *accordion*. It is a collection of clickable fields with titles standing for individual helping subchapters. Then the user selects the name of the chapter, the *accordion* extends the text of the manual to the according chapter. The field is automatically closed when the user clicks on the same title again or a different one. To make the window more navigable, each row is numbered.

- *Dark mode* – a button that changes and inverts the colors of the website.
- *Log out* – a button that is visually separated from the others. By clicking the button, the user is redirected to the login page.

Login and Sign up

The content of the *Login* and *Sign up* consists of the *Header* class and the body. The body of the page is centered in the middle of the page where the input forms are located. The header menu is not clickable until the user is logged in. This was done to obscure empty space and unite the page design since the menu is located on every page of the website.

Home page

The home page is divided into 2 fundamental parts: upper and lower. The first contains the data for displaying the total income and expense for the selected year on the left side. On the right side, there are the *Add* and *Display* buttons, which make it possible to add the data directly from the home page or lead the user to the records page.

The add window contains the following input fields:

1. *Income and Expense*
    - Payment date
    - Type of payment
    - Income or expense
    - Item name
    - Amount
    - Tax creditability
    - Description
2. *Demand and Debt*
    - Date of issue
    - Due date
    - Item name
    - Document number
    - Firm
    - Type
    - Value
    - Tax creditability
    - Description
3. *Assets*
    - Date of enlistment
    - Item name

- Document number
- Purchase price
- Type of payment
- Property type
- Depreciation group
- Method of depreciation
- Description

In the lower part of the home page there is a bar chart displaying the relation between income and expense through the years. I used a JavaScript library *Plot.ly* for the graph. On the right side there is an empty space which will be filled with a pie chart graph, tax calendar or history of added items.

Records

There are 7 *Records* pages which have similar designs: *Income and Expense, Demand and Debt, Fixed assets, Minor assets, Income and Expenditure journal, Bank book* ("Kniha bankovní") and *Cash book* ("Kniha pokladní").

- *Income and Expense* and *Demand and Debt* – these pages have an *Add* button at the top left and next to it, in the middle, there is the title of the page (in this case "Příjmy a výdaje" and "Pohledávky a dluhy"). When the user clicks the *Add* button there appears a pop-up window that contains form fields. At the top, there is the name of the item with the cross on the right which closes the window. At the bottom, there are two buttons: *Save* and *Close*. When the user clicks the *Save* button, the form fields are checked for empty data. If there is an empty field, the user will be alerted. If this is not the case, the item is saved and a green alert "The item [name] has been added successfully" appears above the table.

  Under the button and the title, there is a table. The table is striped and the text in the header and footer of the table is bold to, distinguish the data from the fixed table structure. If there is data overlapping the height of one page, a scrollable bar appears which allows the user to only scroll the table, not the entire page. Furthermore, the header and footer of the table have fixed locations at the top and bottom, to avoid unnecessary scrolling. The header includes the names of the table columns and the footer contains the summary of financial values.

  Moreover, when the user hovers over the table's header, it changes the cursor style to "hand". This function expresses the possibility of scrolling the individual columns.

  Finally, there are blue and red buttons (*Save* and *Delete*), in each row. After the button is clicked and the task is completed successfully, the user will be alerted with a blue or orange alert to signify an edited item ("Item [name] successfully edited") or deleted item ("Item [name] successfully deleted").

  The alert automatically disappears when the user executes another action on the page.

- *Fixed assets* – the page has the same design as the *Income and Expense* and *Demand and Debt* excluding the fixed footer of the table and buttons.

  There are two more buttons in addition to the *Add* and *Edit*. The first is *Eliminate* ("Vyřadit") which is a drop-down button with items *Disposal* ("Likvidace") and *Sale*. When the user clicks the item, the asset is fully depreciated. The information about the asset depreciation is hidden under the *Depreciation* ("Odpisy") button, which displays a pop-up window. In this window, besides the title in the header and a *Close* button in the footer, there is a table that is dynamically generated to display the assets. The table has four columns:

  - *Order* – stands for the row's number.
  - *Value* – how much has been depreciated in the current year.
  - *Remaining value* – how much there is to depreciate.
  - *Year of depreciation* – which year the depreciation was executed.

- *Income and Expenditure journal, Bank book* and *Cash book* – these pages are very similar. First, there is a title on the left side, under which the table is located. The table has the same styling as the previous ones - stationary header and footer. However, in the header besides the names of the columns, there are also two date selectors. The left states the start of a time period and the right states the end.

  The table does not generate database data unless the user clicks on the button *Submit* next to the date selectors. In the *Income and Expenditure journal* it creates an overview of overall data related to the income or the expense. The *Bank book* generates a list of incomes and expenses that are marked as bank transmissions. The *Cash book* in contrast to the *Bank book* generates data related to the payments in cash.

Edit

Edit is the page navigable from the *Income and Expenditure, Demand and Debt* and *Assets* by pressing the *Edit* button.

The page displays the input fields that are available while inserting the data in the *AddIncomeExpense, AddDemandDebt* or *AddAsset* method (4.1.4). They are structured in the middle of the page with the title above, indicating that the user is editing an item. At the bottom, under the input fields, there are two buttons: *Save* and *Cancel*. When the user presses the *Save* button, they are redirected to the records page and alerted by a blue alert with the message "The items [name] has been successfully edited.".

In the *Fixed assets*, the fields have additional JavaScript functionality – the assets can not be edited after they are added. The only allowed fields to edit are *Item name*, *Document number* and *Description* – they do not directly affect the logical part of the application while calculating the depreciation.

The initial idea was to keep the edit pages as pop-up windows the same way as *Add* windows, however, due to complications with the AJAX, this was not pursued.

Delete

The delete is executed when the user clicks on the *Delete* button on any of the *Income and Expense, Demand and Debt* and *Assets* pages. If the button is clicked, a JavaScript *confirm* window appears at the top, to confirm the deletion of the item. After the item is deleted, an orange message alert "The item [name] has been deleted successfully." is displayed above the body of the page.

4.1.5 Application Logic

The application logic consists of the depreciation function and a document number generator.

The depreciation logic is in the *AppLogic* class. This class has one key method *ExecuteDepreciation*, which, among others, includes the functionality of the following methods:

1. *SaveDepreciation*
2. *CheckIfDepreciationThisYear*
3. *IsLastYear*

First, the method accesses the database table *assets* for the logged user and then loads the necessary properties of the asset. This is followed by a cycle that is not finished until the date of acquisition of the asset is earlier than the current year, and at the same time, it is equal to or smaller than the table value for this asset given by the law. Second, it checks if there should be a depreciation with the *CheckIfDepreciationThisYear* method and then it makes a decision based on the depreciation method – *Straight-line* ("Rovnoměrná") or *Accelerated* ("Zrychlená") method.

The methods differ in variable values. In the methods, there are two different arrays to determine the coefficient for the upcoming equation. They are filled with percentages representing the part of the total price which should be subtracted each year. Then, the method decides whether it is the first year of subtraction or not and it uses either the first or the second array. After that, the equation is set to the equations in chapter 1.2.5. The value generated by the equation is added to the *sum* variable, which serves for the calculation of the remaining value. The remaining value is important because it is later displayed in the final column together with the value that has been evaluated from the equation. Then, it either saves the asset depreciation with the method *saveDepreciation* into the *assetdepreciation* table if there is a depreciation the current year. If not, it sets the row as the last year which makes this the last cycle. For the database, it marks the state as last with "1" (boolean variable) and also sets *hiddenSlot* variable as *Eliminated*. It then saves the asset depreciation.

However, the cycle can be suddenly interrupted by the user by clicking the *Eliminate* button on the page. In this case, the previous year's value is set as the remaining value which is now set as zero and last year's state as "1". Now it can be saved as an early elimination in the database and set an alert message "The item [name] successfully eliminated.". This asset, from now on, will not be depreciated.

Finally, this method is executed in the *Check* method in the *Authentificator* class, which checks if the user is logged in. Now, the depreciation will be executed anywhere on the website and at the same time, allows the display of the alert message if the asset has been depreciated after the user has logged in. This method for the calculation of the asset's depreciation is executed when creating the asset as well. This ensures both checking if the asset should be immediately depreciated if the user inserts an older date than one year or is log in to the application.

The second part of the logic is the function for generating the document number for each of the *Records* pages. This is solved by saving the number in the variable using the function *GenerateDocument* described previously.

Chapter 5

Testing and Publication

To enhance the functions and styling, four people were asked to try and test the application. Each person was chosen regarding their experience and knowledge. Their knowledge in this field is as follows:

1. Person 1 has a basic knowledge of programming and little experience with using accounting software. During the trial, they found a few bugs and styling problems. Later, this person did not participate in testing but helped with finding bugs.

2. Person 2 has their own company and an e-shop (Ancer UPS, Ltd.). Furthermore, they manage accounting in an Excel application. In the trial of the application, they recommended a few design features.

3. Person 3 knows PHP on an expert level and has minor experience with accounting. In this application, they have helped with creating two depreciation functions.

4. Person 4 has an accounting company (VTVS, Ltd.). They manage tax records in POHODA software. They provided a look into the application and recommended a few appearance modifications.

5.1 Pre-testing phase

Before testing the application, each individual was asked to try the application and provide initial feedback or comments.

Person 1

The first person was focused on testing the responsiveness, input fields and finding bugs on the website. While testing the fields, they managed to find out that the table columns in each of the records pages would become, after inserting a very long string of data, unlimitedly wide. This made the whole page horizontally scrollable and not responsible for other page elements. The issue was solved by applying a scrollable field on the table instead of the body of the page.

Another problem appeared when sending data to the database. Person 1 uses the browser Mozilla Firefox, which did not send or receive data from the database correctly, i.e. UTF-8 coding did not work as expected with Google Chrome. The issue was solved by using a more explicit meta tag and charset in HTML and PHP for the database connection. This was also tested on the Safari browser and the problem did not occur in that instance.

Ultimately, this person is used to working with an application for creating and editing products, where they operate with multiple products at the same time. They were missing the "Save and Next" button on the page where the item is added, which would simplify the addition of multiple items.

In summary, this person has helped with enhancing the responsibility of the website, found out that different browsers might work with the database data differently, and recommended adding a new button.

Person 2

Person 2 had several comments during the trial of the application regarding the styling of the application.

First, when navigating to the records page, they did not appreciate the absence of the title of the page, since it was not obvious where on the website they were located. Moreover, they realized that the menu at the top of the page was not showing active pages correctly.

Second, on the home page, the selection field of the year displayed a list of years from the year 1900. This seemed very confusing for the person and they recommended only showing years that are saved in the database. Furthermore, they suggested it would be better to display a pie chart for just this year, instead of the bar chart for all years saved in the database.

Furthermore, the graph is plotted with a JavaScript library called *Plotly*, which, by default, generates several functions for the graph where the user can zoom in, reset zoom, or choose between multiple selection fields, which for this person seemed unnecessary and confusing.

Finally, the home page on their monitor did not fit on one page and had to be scrolled. They recommended making the height of the home page responsible for all monitors. However, this would ruin the appearance of the graph itself.

In conclusion, this person helped with creating more navigable pages by adding titles and recommended a different approach to selecting data from the database, the display of a simple graph without additional functions, and better element responsibility on the home page.

Person 3

The third person is an experienced PHP programmer and brought several code-related comments.

First, after I had created the algorithm for calculating an asset's depreciation, the values were calculated correctly, however, the algorithm was missing an essential function to check if the depreciation has been saved in the database for this year. The person suggested making this a separate function, only executed if the conditions were met in the depreciation function.

Second, there was a need to create a function that performs the verification of these conditions. The function would list the database data of the depreciated asset and make a comparison between the current and previous year in the database. This process will efficiently determine when the data has to be saved.

Finally, they recommended saving similar elements of each site in the session variables to make the code shorter.

In summary, Person 3 has helped to create functions for depreciation and recommended ways of making the code shorter.

Person 4

The last person, having the most experience in accounting and tax records, suggested implementing the following points:

- In each table, display the date first to make data more consistent

- Remove tax from fixed assets because there is always tax included
- Add a button in the asset records table to distinguish the reason for the depreciation of each asset (liquidation and sale)
- Add a cash register to the home page which would show the total amount of money available
- Remake the income and expense records page, so it would only show data from the current year
- Change the order of the input fields
- Add an option to select a type of the document in Income and Expense
- Separate tangible and intangible assets as they are depreciated in a different way (intangible asset is not limited by the amount of 80,000 CZK)
- Add a method of acquisition in the Fixed assets
- Keep accounting data so it can be later transferred to the accounting

They also recommended making the application for non-taxpayers when talking about tax in the application. This would otherwise lead to added difficulty functions, would include greater research of accounting laws and require more consultations.

In conclusion, Person 4 recommended adding a few functionalities that can be seen in POHODA software, such as the order of columns or the format of the data for accounting. Due to limited time, some of the points will not be included and will be considered in the future.

5.2 User Acceptance Testing

To ensure the application is functioning correctly, it is user-friendly, and it satisfies functional requirements in the analysis, each person was asked to follow these instructions:

- Go to `www.danovaevidencecepela.cz`
- Create a new account and log in to the application.
- Use a guide located in the top menu under:
  1. "Uživatelské jméno"
  2. "Nápověda"
- Suppose your company bought a laptop named *DELL* valued at 25,000 CZK and started using it immediately. It was paid in cash. Try to insert the item into the correct table.

  Now change the description to "laptop".
- Suppose your company bought a table named *HOBIS* valued at 4700 CZK. Put it in the correct table.
- Suppose this year your company sold a power generator named *NZ UNI* valued at 1,000,000 CZK. The payment was made by bank transfer. Insert the item into the correct table.

- Suppose your company owes 450 CZK for a credit from a company *O2* that has to be paid by November 15th. The amount has already been paid by bank transfer. Insert it into the appropriate table.
- Suppose your company just bought a *Renault Master* for 700,000 CZK on February 13th, 2021 and started using it on the same day. Put it into the correct table. Then, try to locate the depreciation of the car on the same page.
- Suppose the *Renault Master* has crashed and cannot be repaired. Eliminate the car from the records.
- Now, find the tax base for the period from January 1st, 2023, to December 31st, 2023.
- Try to recognize the previously added points on the homepage graph.

After the testing, each of the testers was asked additional questions:

1. Did you have any issues while following the previous instructions, and if so, which exactly?
2. Did you find any bugs or issues with the browser compatibility?
3. Was it easy to navigate on the website?
4. What information did you find helpful on the guide page?
5. Do you have any suggestions for improving the User Interface?
6. Is there anything that you found confusing?
7. Do you have any additional comments?

5.2.1 Feedback

This chapter summarizes each person's experience with the application.

Person 1

This person was the first to see the tasks, and they managed to find a few errors which they were focused on finding.

First, in task 2, they wanted to insert the item as a minor asset in the quick navigation menu on the home page which caused an error with saving the asset – in this application, the minor asset can only be inserted in the income and expense window while they were trying to add it as a fixed asset with a lower value than 80,000. The error was quickly recognized and fixed.

Furthermore, they tried to select a date in the future in the assets inquisition, which caused unexpected calculations. For this reason, adding a future date for the asset is impossible. They also recommended extending the limit of characters that can be sent to the database as there would appear an error if the form input size is too long.

These bugs were acknowledged and fixed before reaching the other persons.

Person 2

The person answered the questions overall positively. They said that there were no issues while following instructions and they did not find any bugs or issues with the browser. Furthermore, they said that the guide and the application are intuitive, and the navigation is simple, which was their main concern when trying the application before testing. Furthermore, they had a few suggestions related to graphics such as the use of darker colors in the journal or the use of a bigger font in the menu.

They managed to follow the test successfully.

Person 3

Person 3 managed to complete the test correctly and without many issues. Since they were not qualified in this field, they needed the manual. They had to close the window each time they tried to add data. They suggested adding a function that would navigate them to the correct part of the manual, or make the manual included in the popover window. In the test, they said there were no issues with compatibility nor did they find any bugs. Furthermore, it was easy for them to navigate and find the information in the manual.

Overall, they liked the application and navigated easily but individual points in the testing were difficult for them.

Person 4

Person 4 was focused on the accounting part of the application. For the third assignment, they decided to insert the power generator as a demand. This was unexpected and led to finding a bug. The issue was in saving the demand as an income once it is marked as *Compensated* ("Uhrazeno"). However, this was immediately resolved after the testing.

Moreover, in task 5, they explained that a fixed asset should be saved each time as a non-taxable item and individual depreciation should be taxable. For this reason, the option to choose if the asset is taxable or not was removed.

In task 7, when trying to find the tax base, they were not sure how to calculate it, since the homepage includes the information about income and expense value for the selected year but the taxable and non-taxable values were mixed. They suggested separating the values on the home page into taxable and non-taxable. On the page *Income and Expense* ("Příjmy a výdaje") those values can be found regardless of the year acquisition. In future, this page might be updated to have a date selector which would only display the data for the selected year. This will provide easier navigation for the user.

Overall, they found the application intuitive.

5.3 Publication

In this chapter, it is explained how I published the application on the Internet. The provider I have chosen for the domain, web hosting, and database is `wedos.cz`. The application can be found under the domain `www.danovaevidencecepela.cz` and it's code in the `https://github.com/Martyneqq/bc` GitHub repository. The procedure of publishing the application was done by following a tutorial.[21]

To publish the application, I followed the instructions:

1. Obtain a unique domain
2. Purchase a web hosting server

3. Create a database and link it with the server
4. Secure the website
5. Get an application or an extension for managing a file transport to the server

First, I have created a domain. On wedos.cz, this is done by navigating to the "Domains" in the top menu and then selecting the option "Register a domain". A page with a form will pop up and require a domain name which I have set to "danovaevidencecepela", a suffix of the site set as ".cz" (this has a substantial influence on the price), the length of the registration set as 1 year. After this, the Confirm Terms page showed up and I was redirected to the Sign-up page. Now, all the access information and payment requests for the web hosting server and FTP connection were sent to this registered mail address. The domain is active from now on and is displayed in the "List of domains" located in the menu on the left.

Second, the web hosting server where all the files defining the website would be located, is found under "Web hosting" in the top menu. The process is very similar to obtaining the domain. I selected the button "New web hosting" in the menu on the left. Then, I was redirected to the page with basic information about the server. I inserted the domain name "danovaevidencecepela", which linked the server to the domain. Then I selected a variant "Low-cost" which offers 10 GB of disk space, unlimited data transfer, 256 PHP memory limit of 1 GB space for database-related data and HTTPS.[43] In additional services, I selected HTTPS to make the website secure. Then I selected a version of PHP 8.1. Other options were left unchanged. Furthermore, I filled the documents with personal information and confirmed the terms. The server was active after approximately thirty minutes and information about it can be found under "Web hosting" in the top menu and "List of Web hosting" in the menu on the left.

To create a database, I navigated to the table located in the link named "List of web hosting". On this page, five tables can be seen and the second one is the database created by clicking on the hyperlink "Create a new". Then, I inserted the name of the database "evidence", selected the only accessible database type MariaDB and inserted my e-mail adress where the login information would be sent. By clicking on the "Create" button, the database was successfully created and displayed in the table. The link to the database manager is hidden under the database hyperlink where *pma.wedos.net* can be found.

After I had gone through all those steps, the website was successfully displayed on the Internet. However, when purchasing a web hosting server, I selected the option to include HTTPS and it did not activate by default. I navigated to the "HTTPS" item menu on the left. On the page, I selected the option "HTTPS with the Let's Encrypt certificate" and pressed the "Submit" button. Then, after being redirected back to the "HTTPS" page, I pressed the "Add domain" button above the table and selected the domain. This process linked the domain to the HTTPS. After a while, the website finally displayed the secure status.

Finally, when the domain and the web hosting server were both active, the provider sent the user data to connect to the FTP server. The procedure of setting up the FTP is mentioned in the chapter 3.1. By executing these steps, I published this application on the Internet.

In conclusion, in this chapter, I explained who the provider of this web application is, how to create a domain and a web server for this specific provider, where the website can be found, how to properly create the database, how to secure the site and how to set up an FTP connection.

Conclusion

The goal of this bachelor project was to accomplish the functional and non-functional requirements given at the assignment. It was fulfilled and the application is usable for non-payers of VAT to keep tax records in the Czech Republic.

The design of the application is created in CSS and Bootstrap. CSS was used for HTML styling, page structure or sticky elements in the table while Bootstrap is used for creating the buttons, tables and menu. Connection to the database is established by SQL, the structure of the application and algorithms are written in PHP, dynamic elements that do not require page refresh are written in JavaScript, and the markup language used for this application is HTML.

The code of the application is structured in classes and files. It is separated by database access, HTML and application logic. JS and CSS are kept as separate files which increases the readability of the code. Each input form in the application is secured to avoid SQL injections. Both the database and the application use UTF-8 encoding for correct data output from the database. It is suitable for all devices and is fully responsible.

The application was tested by four people with different backgrounds. It provided valuable information on what to improve. Person 1 managed to find a few bugs and compatibility issues with other browsers since they were focused on finding them. Person 2 was focused on improving the design, as it was difficult to use the application. They suggested several user-friendly designs for the application such as larger font and different colours for the text. Person 3 helped with creating a few key functions, in particular, editing the item and asset depreciation. During the testing, they recommended adding a guide for each page to make it easier to understand the application functions. Person 4 provided expert consultation and delivered highly valuable advice on taxes, which included adding a page with the Cashbook and the Bankbook, and adjusting taxes for fixed assets. The consultations with them proved very useful because studying the law and analyzing similar applications were not sufficient for the final version of this application.

The application can be extended in the future. This might include creating a user profile, adding a pie chart or tax calendar, or connecting the application with the *ARES* database to ensure the correct insertion of the person identification number when creating an account. It might also include allowing item filtering, keeping the data on multiple pages, or adding the possibility to include multiple items under one document number.

The application is available to visit at `https://danovaevidencecepela.cz` and has been published using `https://wedos.cz` provider. The code is available at `https://github.com/Martyneqq/bc`.

Bibliography

[1] MINISTERSTVO FINANCÍ ČESKÉ REPUBLIKY. *Vymezení pojmu drobný, malý a střední podnikatel a postup pro zařazování podnikatelů do jednotlivých kategorií*. [online] [Accessed on 2021-12-17]. Available from: https://www.mpo.cz/assets/dokumenty/27300/28024/310086/priloha017.pdf.

[2] Zákon o účetnictví (úplné znění). Předpis č. 563/1991 Sb. ve znění platném od 1. 1. 2022.

[3] Uvírová, L. (2013) *Jak vést Daňovou EVIDENCI?*, Informace pro účetní a podnikatele. Available at: https://portal.pohoda.cz/pro-podnikatele/uz-podnikam/jak-vest-danovou-evidenci/?feed=podnikatel (Accessed: 17 December 2023).

[4] Oliver Leiss, Jasmin Schmidt: *PHP v praxi*. Praha: Grada Publishing 2010. ISBN 978-80-247-3060-8.

[5] Pavel Štohl, Vladislav Klička: *Maturitní okruhy z účetnictví 2022*. Znojmo 2010.

[6] Obecné Informace (2021) *Finanční správa České republiky*. Available at: https://www.financnisprava.cz/cs/dane/dane/dan-z-prijmu/pausalni-dan/obecne-informace (Accessed: 17 December 2023).

[7] Seyfor, Plc. (2023) *Co Je Daňová evidence včetně příkladů + vzor, MoneyS3*. Available at: https://money.cz/novinky-a-tipy/ucetnictvi-2/co-je-danova-evidence-vcetne-prikladu-vz (Accessed: 17 December 2023).

[8] Zapletalová, M. (2023) *Odpisy Majetku v Daňové Evidenci: Jak Je Spočítat*, iDoklad online fakturace. Available at: https://www.idoklad.cz/blog/odpisy-majetku-v-danove-evidenci-jak-je-spocitat (Accessed: 17 December 2023).

[9] Divinová, V. (2021) *Rozdíl Mezi účetními a daňovými odpisy*, Grantex.cz. Available at: https://grantex.cz/blog/rozdil-mezi-ucetnimi-a-danovymi-odpisy/ (Accessed: 17 December 2023).

[10] Šperl, Ing.J. (2022) *Drobný (dlouhodobý) hmotný A Nehmotný Majetek*, Živnostník profi. Available at: https://www.zivnostnik.cz/33/drobny-dlouhodoby-hmotny-a-nehmotny-majetek-un pd-ViQ/ (Accessed: 17 December 2023).

[11] Kandlerová, K. (2015) *Odpisy Majetku V praxi*, Informace pro účetní a podnikatele. Available at: https://portal.pohoda.cz/dane-ucetnictvi-mzdy/ucetnictvi/odpisy-majetku-v-praxi/ (Accessed: 17 December 2023).

[12] Seyfor, Plc. (2021) *Co Je to peněžní Deník a jak ho vyplnit*, MoneyS3. Available at: https://money.cz/novinky-a-tipy/ucetnictvi-2/co-je-to-penezni-denik-a-jak-ho-vyplnit/ (Accessed: 17 December 2023).

[13] STORMWARE, *Srovnání variant*. Available at: https://www.stormware.cz/pohoda/varianty/ (Accessed: 17 December 2023).

[14] Ceník (2022). *iDoklad online fakturace*. Available at: https://www.idoklad.cz/cenik (Accessed: 17 December 2023).

[15] Fakturace online. *iDoklad online fakturace*. Available at: https://www.idoklad.cz/vlastnosti/vystavovani-faktur (Accessed: 17 December 2023).

[16] Accost, Ltd. Účtovna. Available at: https://www.uctovna.cz/ (Accessed: 17 December 2023).

[17] Direct Accounting, Ltd. iÚčto.cz. Available at: https://www.iucto.cz/ (Accessed: 17 December 2023).

[18] W3Schools.com. *HTTPS Methods*. Available at: https://www.w3schools.com/tags/ref_httpmethods.asp (Accessed: 17 December 2023).

[19] W3Schools.com. *HTML Introduction*. Available at: https://www.w3schools.com/html/html_intro.asp (Accessed: 17 December 2023).

[20] Charles M. N. W3C (2017). *HTML 5.2 is done, HTML 5.3 is coming*. Available at: https://www.w3.org/blog/2017/html-5-2-is-done-html-5-3-is-coming/ (Accessed: 17 December 2023).

[21] Berka M. (2023). *Jak vytvořit webové stránky 2024 Tvorba webu*. Available at: https://www.youtube.com/watch?v=rUuwtclaxEM&ab_channel=MaxBerka

[22] MDN Web Docs. *CSS: Cascading Style Sheets*. Available from: https://developer.mozilla.org/en-US/docs/Web/CSS (Accessed: 17 December 2023).

[23] W3Schools.com. *Bootstrap 3 Tutorial*. Available at: https://www.w3schools.com/bootstrap/ (Accessed: 18 December 2023).

[24] GetBootstrap.com. *Build fast, responsive sites with Bootstrap*. Available at: https://getbootstrap.com/ (Accessed: 18 December 2023).

[25] PHP.net. *PHP 8.3.0 Released!*. Available at: https://www.php.net/archive/2023.php#2023-11-23-2 (Accessed: 18 December 2023).

[26] MDN Web Docs. *What is JavaScript?* Available at: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript (Accessed: 18 December 2023).

[27] jQuery.com. *hat is jQuery?* Available at: https://jquery.com/ (Accessed: 18 December 2023).

[28] JSON.org. *Introducing JSON*. Available at: https://www.json.org/json-en.html (Accessed: 18 December 2023).

[29] Štráfelda J. *Ajax*. Available at: https://www.strafelda.cz/ajax (Accessed: 18 December 2023).

[30] AMAZON WEB SERVICES. *What is SQL?* Available at: https://aws.amazon.com/what-is/sql/ (Accessed: 18 December 2023).

[31] phpMyAdmin.net. *Bringing MySQL to the web*. Available at: https://www.phpmyadmin.net/ (Accessed: 18 December 2023).

[32] Apache Friends. *What is XAMPP?*. Available at: https://www.apachefriends.org/ (Accessed: 18 December 2023).

[33] Behrouz A. F. (2000). *FTP* (p. 672). Available at: https://vaibhav2501.files.wordpress.com/2012/02/tcp_ip-protocol-suite-4th-ed-b-forouzan-mcgraw-hill-2010-bbs.pdf (Accessed: 18 December 2023).

[34] Raddy (2020). *How to use FTP Simple in Visual Studio Code - Connect, Edit, Upload*. Available at: https://www.youtube.com/watch?v=G1R5H6pnOyc&ab_channel=Raddy (Accessed: 18 December 2023).

[35] GitHub Docs. *Collaborative coding*. Available at: https://docs.github.com/en (Accessed: 18 December 2023).

[36] Kodousková B. (2021). *Co je HTTPS?*. Available at: https://www.rascasone.com/cs/blog/co-je-https-http-ssl-tls (Accessed: 18 December 2023).

[37] Postal C. (2023). *What is Cryptographic Key Exchange?* Available at: https://www.upguard.com/blog/diffie-hellman (Accessed: 18 December 2023).

[38] Gillis, A.S. (2022) *What is Diffie-Hellman key exchange?: TechTarget, Security*. Available at: https://www.techtarget.com/searchsecurity/definition/Diffie-Hellman-key-exchange (Accessed: 02 January 2024).

[39] Buchmann J. A. (2013) *Introduction to Cryptography*. Available at: https://books.google.cz/books?id=BuQlBQAAQBAJ&pg=PA190&redir_esc=y#v=onepage&q&f=false (Accessed: 19 December 2023).

[40] W3Schools.com. *HTTP Request Methods*. Available at: https://www.w3schools.com/tags/ref_httpmethods.asp (Accessed: 18 December 2023).

[41] OWASP.org. *SQL Injection Prevention Cheat Sheet*. Available at: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html (Accessed: 18 December 2023).

[42] Freda A. Avast, Ltd. *What Is the MD5 Hashing Algorithm and How Does It Work?*. Available at: https://www.avast.com/c-md5-hashing-algorithm (Accessed: 18 December 2023).

[43] WEDOS, Plc. *Webhosting LowCost*. Available at: https://www.wedos.com/cs/webhosting/lowcost/ (Accessed: 18 December 2023).

[44] W3Schools.com. *How TO - Sort a List*. Available at: https://www.w3schools.com/howto/howto_js_sort_list.asp (Accessed: 18 December 2023).

[45] Plotly.com. *Bar Charts in JavaScript*. Available from: https://plotly.com/javascript/bar-charts/ (Accessed: 17 December 2023).

[46] jQuery.com. *jQuery JavaScript Library v3.6.1*. Available from: https://jquery.com/ (Accessed: 20 December 2023).

Appendix A

Uživatelská příručka

Zde najdete návod pro užití aplikace.

A.1 Vytvoření uživatelského účtu

Na stránce pro přihlášení kliknete na odkaz "Vytvořit nový účet". Pro registraci je nutné vyplnit uživatelské jméno, identifikační číslo osoby, email a heslo.

Zadané heslo musí obsahovat alespoň 8 znaků, alespoň jednu číslici a jedno písmeno.

A.2 Přihlášení

Pro přihlášení zadáte uživatelské jméno a heslo zvolené při registraci. Po vytvoření účtu budete automaticky přesměrování na stránku s přihlášením.

A.3 Odhlášení

Pro odhlášení klikněte na své uživatelské jméno v horním menu a zvolíte možnost "Odhlasit se".

A.4 Zobrazení celkových příjmů a výdajů pro zvolený rok

Na domovské stránce kliknete na seznam roků. Každý rok reprezentuje hodnotu příjmu nebo výdaje v databázi. Vyberete rok, ze kterého chcete zobrazit součet příjmů a výdajů a kliknete na tlačítko "Potvrdit". Dole se poté zobrazí celková hodnota pro příjmy a výdaje zvlášť.

A.5 Evidence příjmů a výdajů

Evidence příjmů a výdajů se nachází v horním menu:

1. Evidence
2. Deník příjmů a výdajů

Deník zobrazuje informace o přidaných příjmech a výdajích, které jsou v tabulce seřazené podle data. Hodnoty příjmů a výdajů v tabulce jsou dále rozděleny na daňové/nedaňové podle toho, zda uživatel při přidávání zvolil daňovou uznatelnost nebo ne. Součet těchto hodnot se nachází v dolní části tabulky. Pokud se v databázi nachází odpis nebo drobný majetek, řádek tabulky, kde se nachází název položky bude změněn na "Odpis: [název]" nebo "Drobný majetek: [název]".

Jednotlivé sloupce obsahují:

- Datum uhrazení - datum zavedení položky do evidence
- Číslo dokladu - tento rok + typ dokladu + pořadí (například 23HV3 - v tomto případě se jedná o 3. doklad vystavený v tomto účetním období, placený hotově a jedná se o výdaj)
- Název - název položky
- Příjem nebo výdaj - zda se jedná o příjem či výdaj
- Daňové/nedaňové příjmy - příjmy zahrnující/nezahrnující daň
- Daňové/nedaňové výdaje - výdaje zahrnující/nezahrnující daň
- Způsob platby - zda platba proběhla v hotovosti nebo převedením na účet
- Popis - libovolná poznámka s limitem 128 znaků
- Úpravy - operace pro upravení/smazání řádku

Příklad: prodali jste čtvrtou elektrocentrálu "NZ UNI" za toto účtovací období v den 4.10.

Datum uhrazení	Číslo dokladu	Název	Příjem nebo výdaj	Daňové příjmy	Způsob platby
04/10	23HV4	NZ UNI	Příjem	4,700.00	Z účtu

Table A.1: Příklad vyplnění tabulky

A.6 Evidence pohledávek a závazků

Pohledávky a závazky se nachází v horním menu pod:

1. ->Evidence
2. ->Pohledávky a závazky

Jednotlivé sloupce obsahují:

- Datum uhrazení - den, kdy částka musí být uhrazena
- Datum splatnosti - do kdy je potřeba částku uhradit
- Číslo dokladu - tento rok + typ dokladu + pořadí (například 23HV3 - v tomto případě se jedná o 3. doklad vystavený v tomto účetním období, placený hotově a jedná se o výdaj)
- Název - název položky
- Firma - po kom je pohledávka vymáhána nebo komu splatit dluh
- Typ - pohledávka nebo dluh

- Hodnota - pohledávaná nebo dlužená částka
- Daňová uznatelnost - zda hodnota zdanitelná či nikoli
- Popis - libovolná poznámka s limitem 128 znaků
- Úpravy - operace pro upravení/smazání řádku

Příklad: firma Vodafone vám poskytuje měsíční kredit na firemní telefon a máte zaplatit 556 CZK. Tímto měsícem (říjen) se jedná o desátou fakturu přijatou.

Datum vystavení	Číslo dokladu	Název	Firma	Typ	Hodnota	Daňová uznatelnost	Uhrazeno
03/10/2023	23BV10	Kredit	Vodafone	Dluh	556,00	Ano	Ano

Table A.2: Příklad vyplnění tabulky

Pozn.: Pokud je sloupec "Uhrazeno" označeno jako "Ano", pohledávka nebo dluh bude uložen jako příjem nebo výdaj. Po této změně položka nemůže být upravena ani smazána.

A.7 Evidence majetku drobného a dlouhodobého

Dlouhodobý a drobný majetek se nachází v horním menu pod:

1. ->Majetek
2. ->Dlouhodobý majetek/->Drobný majetek

Dlouhodobý hmotný majetek je majetek firmy, který přesahuje hodnotu 80 000 CZK. Výjimka nastává v případě, kdy je majetek nehmotný - ten může mít hodnotu menší a je také považován za dlouhodobý majetek, nikoli drobný. Dlouhodobý majetek se každý rok automaticky odepisuje a seznam odpisů najdete pod tlačítkem "Odpisy". Každý odpis se dále eviduje v Evidenci příjmů a výdajů jako výdaj a najdete ho pod názvem "Odpis: [název]". Drobný majetek se automaticky ukládá jako výdaj, jelikož podle zákona nesmí přesahovat hodnotu 80 000 CZK, a najdete ho pod názvem "Drobný majetek: [název]" v Evidenci příjmů a výdajů.

Pokud je dlouhodobý majetek přidán, můžete upravovat pouze název, doklad a popis. Pokud jste při vytváření majetku například zadali špatnou cenu, je nutné majetek smazat a vytvořit znovu.

Jednotlivé sloupce obsahují:

- Datum zařazení - den užívání majetku ve firmě
- Datum vyřazení - předpokládaný den vyřazení majetku
- tento rok + typ dokladu + pořadí (například 23HV3 - v tomto případě se jedná o 3. doklad vystavený v tomto účetním období, placený hotově a jedná se o výdaj)
- Název - název položky
- Počáteční hodnota - nákupní cena majetku
- Typ - majetek hmotný nebo nehmotný
- Odpisová skupina - odpisové skupiny jsou rozděleny podle čísel (1-6), které můžete libovolně zvolit podle toho, kolik let chcete majetek odepisovat s jakou hodnotou.

- Způsob odpisu - rovnoměrný nebo zrychlený. Můžete si vybrat tu metodu, která vám vyhovuje více. Pro dražší majetek se doporučuje užívat rovnoměrný odpis. Procenta, která se odečtou z ceny majetku každý rok jsou uvedena v sekci "Odepisování majetku".

- Popis - libovolná poznámka s limitem 128 znaků

- Úpravy - operace pro upravení/smazání řádku

Příklad: koupili jste na firmu auto Škoda Octavia 2 za 290 000 CZK dne 19.10.2023, vybrali jste, že ho chcete odepisovat po dobu 5 let (odpisová skupina 2) a rovnoměrnou metodou.

Zařazení	Vyřazení	Doklad	Název	Částka	Typ	Skupina	Odpis
19/10/2023	19/10/2028	23BV2	Škoda Octavia 2	290,000.00	Hmotný	2	Rovnoměrný

Table A.3: Příklad vyplnění tabulky dlouhodobého majetku

A.8 Deník

Deník příjmů a výdajů najdete pod tlačítkem "Deníky" v horním menu. Deník vypočítá částku z příjmů a výdajů potřebnou pro určení základu daně v daném období.

Knihu pokladní najdete pod tlačítkem "Deníky" v horním menu. Tento deník vyhledává všechny položky uhrazeny hotově a zobrazí je na základě zvoleného datumu.

Knihu bankovní najdete pod tlačítkem "Deníky" v horním menu. Tento deník vyhledává všechny položky uhrazeny bankovním převodem a zobrazí je na základě zvoleného datumu.

Vypočítaná čáska základu daně pro každou tabulku se nachází ve spodní části tabulky. Toto období zvolíte pomocí polí s datumy - první pole je začátek a druhé konec období. Pokud se v databázi nachází odpis, vyřazení majetku nebo drobný majetek, řádek tabulky, kde se nachází název položky bude změněn na "Odpis: [název]", "Vyřazení: [název]" nebo "Drobný majetek: [název]".

A.9 Vyřazení majetku

Majetek vyřadíte pomocí rozbalovacího tlačítka "Vyřadit" kliknutím na možnosti "Likvidace/Prodej/Darování". Podle zvolené možnosti se přizpůsobí datum vyřazení. Odepsaná hodnota vyřazeného majetku bude zobrazena pod tlačítkem "Odpis" a v Evidenci příjmů a výdajů.

A.10 Odepisování majetku

Majetek se každý rok odepisuje automaticky. Jednotlivé odpisy najdete v Dlouhodobém majteku pod tlačítkem "Odpisy". Následující tabulka udává příklady majetků, které by mohli patřit do jedné ze skupin:

- *a* udává procento odečtené ze vstupní ceny v prvním roce u odpisu rovnoměrného

- *b* udává procento odečtené ze zbylé hodnoty ve zbylých letech u odpisu rovnoměrného

- *x* udává procento odečtené ze vstupní ceny v prvním roce u odpisu zrychleného

- *y* udává procento odečtené ze zbylé ceny ve zbývajících letech ve odpisu zrychleného

Skupina	Příklad	Roky	a	b	x	y
1	Počítač, levné nástroje	3	20	40	3	4
2	Auto, autobus, další nástroje	5	11	22,25	5	6
3	Parní kotle, tramvaje	10	5,5	10,5	10	11
4	Budovy, potrubí	20	2,15	5,15	20	21
5	Průmyslové stavby, byty, zemědělství	30	1,4	3,4	30	31
6	Administrativní budovy, obchodní domy, hotely	50	1,02	2,02	50	51

Table A.4: Příklady majetků s předpokládanou dobou odpisů a ročními procenty

A.11 Řazení tabulek

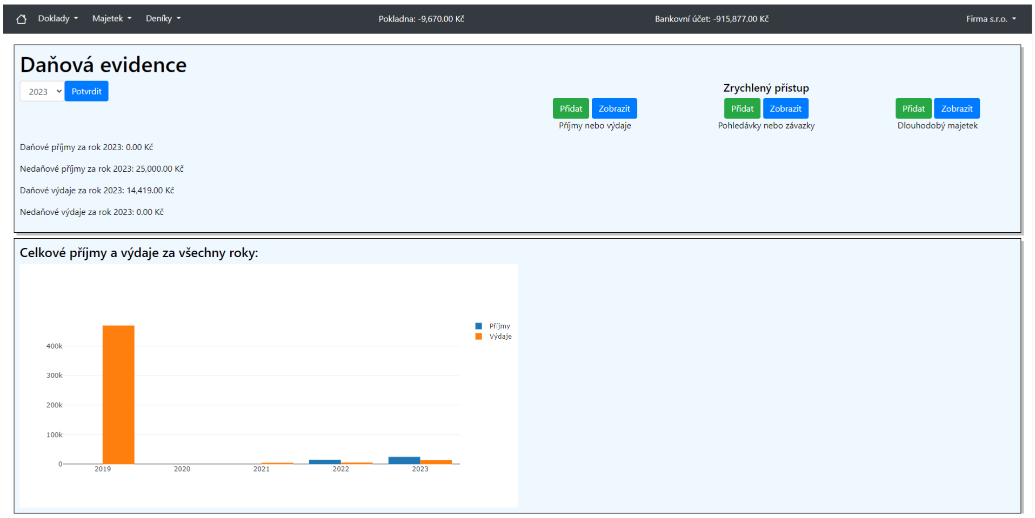
Sloupce tabulek můžou být seřazeny po kliknutí na hlavičku příslušné tabulky. Po opětovném kliknutí se tabulka seřadí v opačném pořadí.

A.12 Typy dokumentů

- **HP** – Hotovostní příjem
- **HV** – Hotovostní výdaj
- **BP** – Bankovní příjem
- **BV** – Bankovní výdaj

Appendix B

Screenshots from the application



Pokladna: -9,670.00 Kč

Bankovní účet: -915,877.00 Kč

Daňová evidence

2023

Daňové příjmy za rok 2023: 0.00 Kč

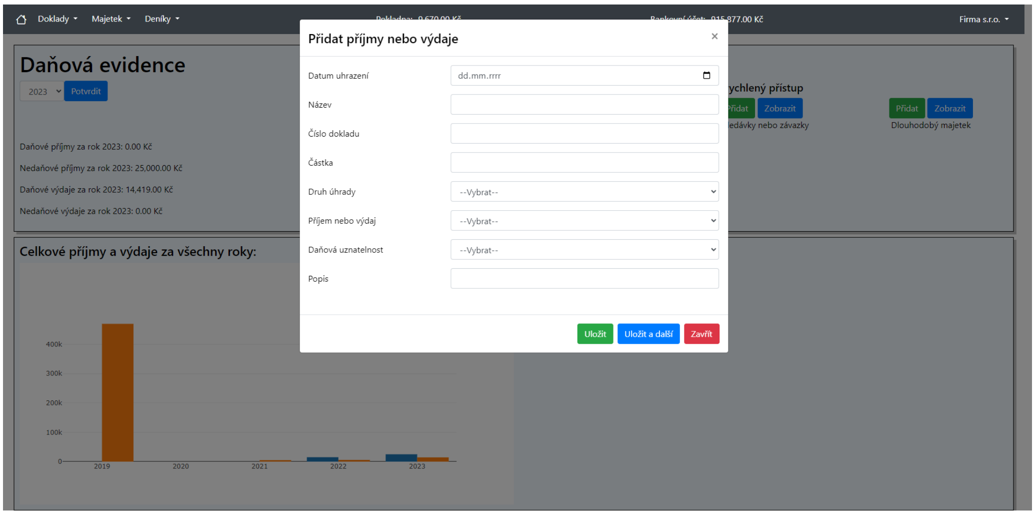
Nedaňové příjmy za rok 2023: 25,000.00 Kč

Daňové výdaje za rok 2023: 14,419.00 Kč

Nedaňové výdaje za rok 2023: 0.00 Kč

Celkové příjmy a výdaje za všechny roky:

(a) Application’s home page.



Přidat příjmy nebo výdaje

Datum uhrazení: dd.mm.rrrr

Název:

Číslo dokladu:

Částka:

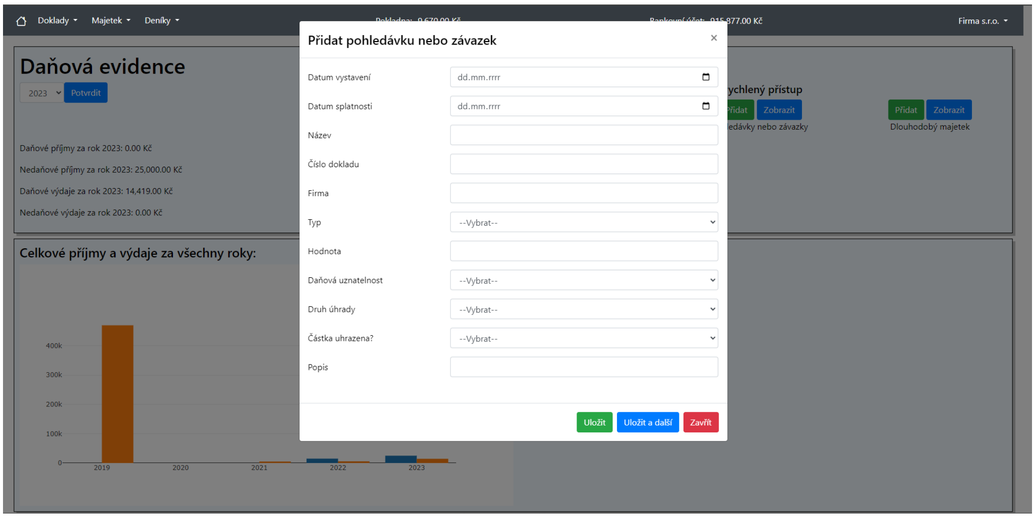
Druh úhrady: --Vybrat--

Příjem nebo výdaj: --Vybrat--

Daňová uznatelnost: --Vybrat--

Popis:

(b) A pop-up window for adding income or expense.



Daňová evidence

2023

Daňové příjmy za rok 2023: 0.00 Kč

Nedaňové příjmy za rok 2023: 25,000.00 Kč

Daňové výdaje za rok 2023: 14,419.00 Kč

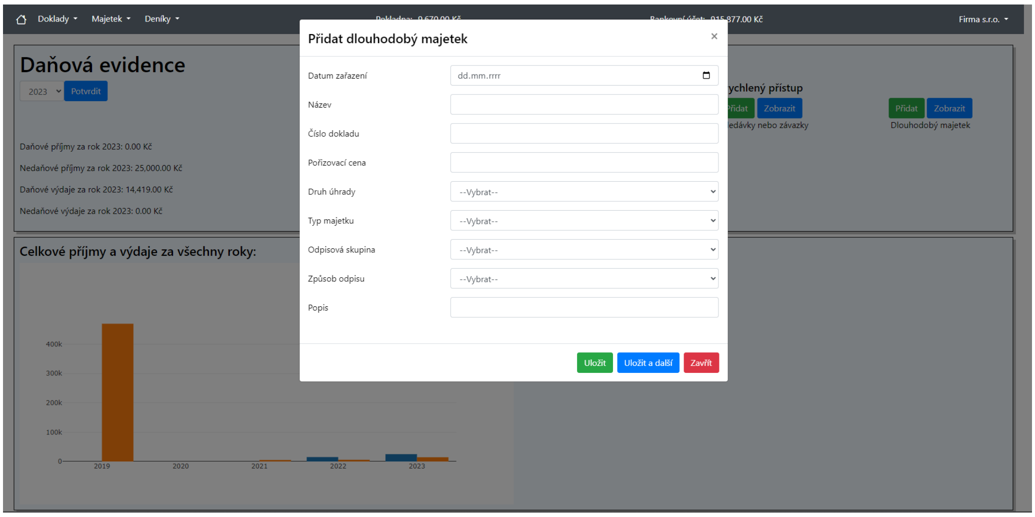
Nedaňové výdaje za rok 2023: 0.00 Kč

Celkové příjmy a výdaje za všechny roky:

Přidat pohledávku nebo závazek

Datum vystavení	dd.mm.rrrr
Datum splatnosti	dd.mm.rrrr
Název	
Číslo dokladu	
Firma	
Typ	--Vybrat--
Hodnota	
Daňová uznatelnost	--Vybrat--
Druh úhrady	--Vybrat--
Částka uhrazena?	--Vybrat--
Popis	

(c) A pop-up window for adding demand or debt.



Daňová evidence

2023

Daňové příjmy za rok 2023: 0.00 Kč

Nedaňové příjmy za rok 2023: 25,000.00 Kč

Daňové výdaje za rok 2023: 14,419.00 Kč

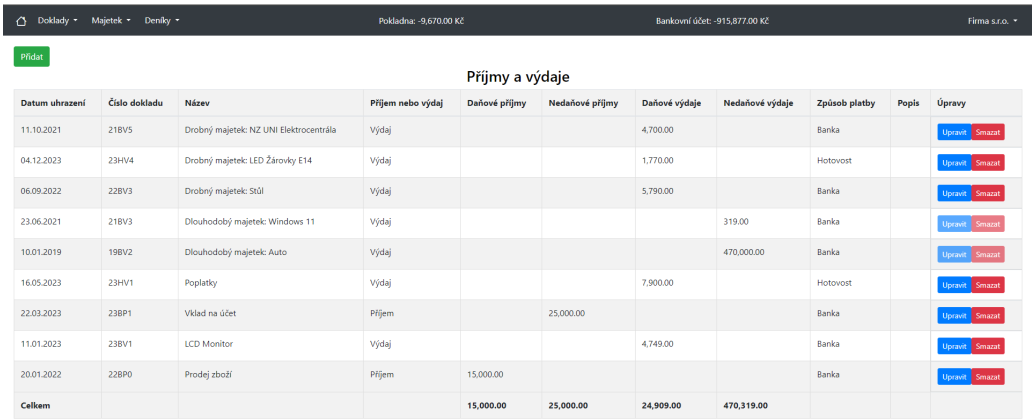
Nedaňové výdaje za rok 2023: 0.00 Kč

Celkové příjmy a výdaje za všechny roky:

Přidat dlouhodobý majetek

Datum zařazení	dd.mm.rrrr
Název	
Číslo dokladu	
Pořizovací cena	
Druh úhrady	--Vybrat--
Typ majetku	--Vybrat--
Odpisová skupina	--Vybrat--
Způsob odpisu	--Vybrat--
Popis	

(d) A window for adding fixed assets.



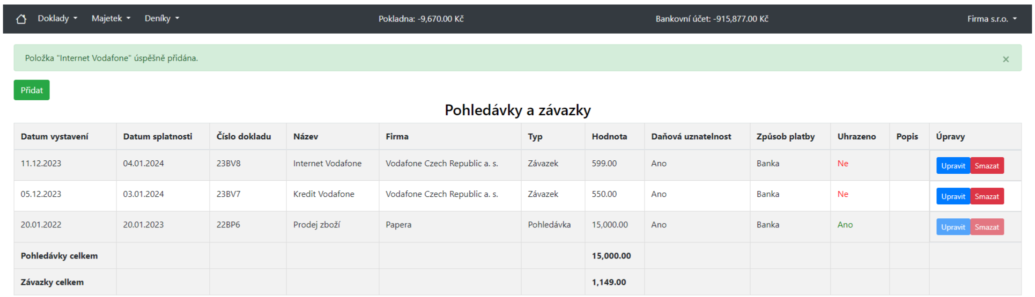
Pokladna: -9,670.00 Kč

Bankovní účet: -915,877.00 Kč

Příjmy a výdaje

Datum uhrazení	Číslo dokladu	Název	Příjem nebo výdaj	Daňové příjmy	Nedaňové příjmy	Daňové výdaje	Nedaňové výdaje	Způsob platby	Popis	Úpravy
11.10.2021	21BV5	Drobný majetek: NZ UNI Elektrocentrála	Výdaj			4,700.00		Banka		Upravit Smazat
04.12.2023	23HV4	Drobný majetek: LED Žárovky E14	Výdaj			1,770.00		Hotovost		Upravit Smazat
06.09.2022	22BV3	Drobný majetek: Stůl	Výdaj			5,790.00		Banka		Upravit Smazat
23.06.2021	21BV3	Dlouhodobý majetek: Windows 11	Výdaj				319.00	Banka		Upravit Smazat
10.01.2019	19BV2	Dlouhodobý majetek: Auto	Výdaj				470,000.00	Banka		Upravit Smazat
16.05.2023	23HV1	Poplatky	Výdaj			7,900.00		Hotovost		Upravit Smazat
22.03.2023	23BP1	Vklad na účet	Příjem		25,000.00			Banka		Upravit Smazat
11.01.2023	23BV1	LCD Monitor	Výdaj			4,749.00		Banka		Upravit Smazat
20.01.2022	22BP0	Prodej zboží	Příjem	15,000.00				Banka		Upravit Smazat
Celkem				**15,000.00**	**25,000.00**	**24,909.00**	**470,319.00**			

(e) A page with records of incomes and expenses.



Pokladna: -9,670.00 Kč

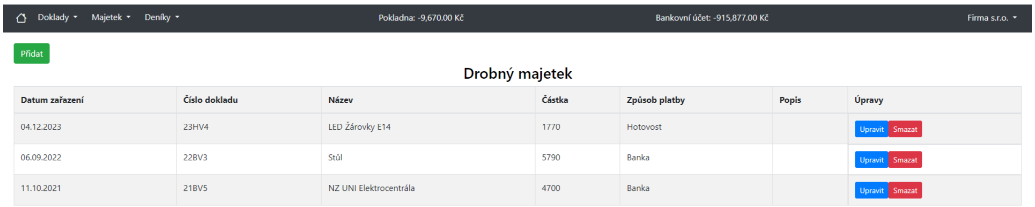
Bankovní účet: -915,877.00 Kč

Položka "Internet Vodafone" úspěšně přidána. ×

Pohledávky a závazky

Datum vystavení	Datum splatnosti	Číslo dokladu	Název	Firma	Typ	Hodnota	Daňová uznatelnost	Způsob platby	Uhrazeno	Popis	Úpravy
11.12.2023	04.01.2024	23BV8	Internet Vodafone	Vodafone Czech Republic a. s.	Závazek	599.00	Ano	Banka	Ne		Upravit Smazat
05.12.2023	03.01.2024	23BV7	Kredit Vodafone	Vodafone Czech Republic a. s.	Závazek	550.00	Ano	Banka	Ne		Upravit Smazat
20.01.2022	20.01.2023	22BP6	Prodej zboží	Papera	Pohledávka	15,000.00	Ano	Banka	Ano		Upravit Smazat
Pohledávky celkem						**15,000.00**					
Závazky celkem						**1,149.00**					

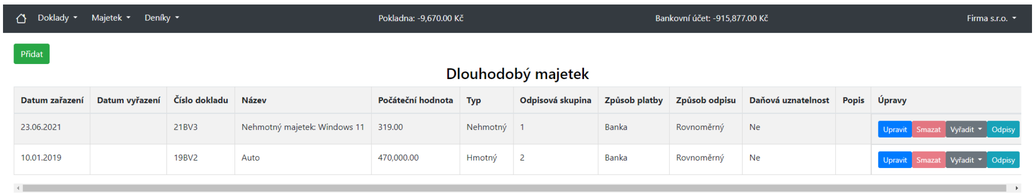
(f) A page with records of demand and debt.



Drobný majetek

Datum zařazení	Číslo dokladu	Název	Částka	Způsob platby	Popis	Úpravy
04.12.2023	23HV4	LED Žárovky E14	1770	Hotovost		Upravit Smazat
06.09.2022	22BV3	Stůl	5790	Banka		Upravit Smazat
11.10.2021	21BV5	NZ UNI Elektrocentrála	4700	Banka		Upravit Smazat

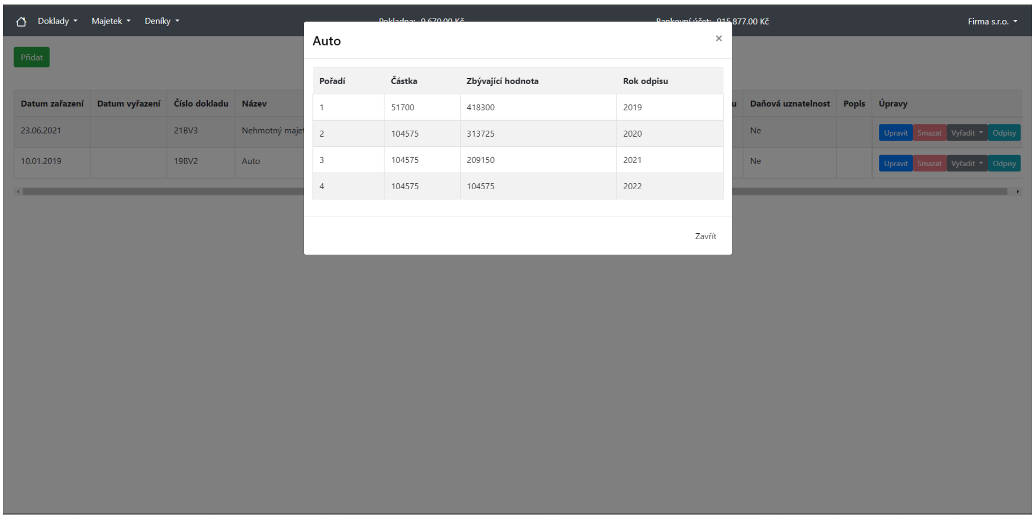
(g) A page with records of the minor assets.



Dlouhodobý majetek

Datum zařazení	Datum vyřazení	Číslo dokladu	Název	Počáteční hodnota	Typ	Odpisová skupina	Způsob platby	Způsob odpisu	Daňová uznatelnost	Popis	Úpravy
23.06.2021		21BV3	Nehmotný majetek: Windows 11	319.00	Nehmotný	1	Banka	Rovnoměrný	Ne		Upravit Smazat Vyřadit Odpisy
10.01.2019		19BV2	Auto	470,000.00	Hmotný	2	Banka	Rovnoměrný	Ne		Upravit Smazat Vyřadit Odpisy

(h) A page with records of fixed assets

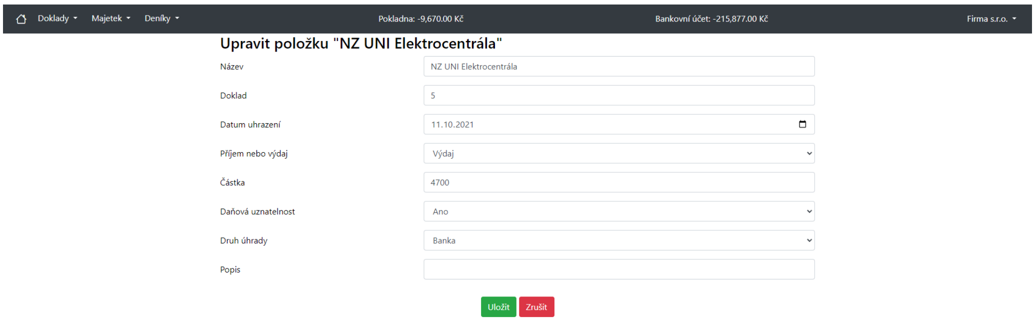


Datum zařazení	Datum vyřazení	Číslo dokladu	Název	Daňová uznatelnost	Popis
23.06.2021		21BV3	Nehmotný maje	Ne	
10.01.2019		19BV2	Auto	Ne	

Auto

Pořadí	Částka	Zbývající hodnota	Rok odpisu
1	51700	418300	2019
2	104575	313725	2020
3	104575	209150	2021
4	104575	104575	2022

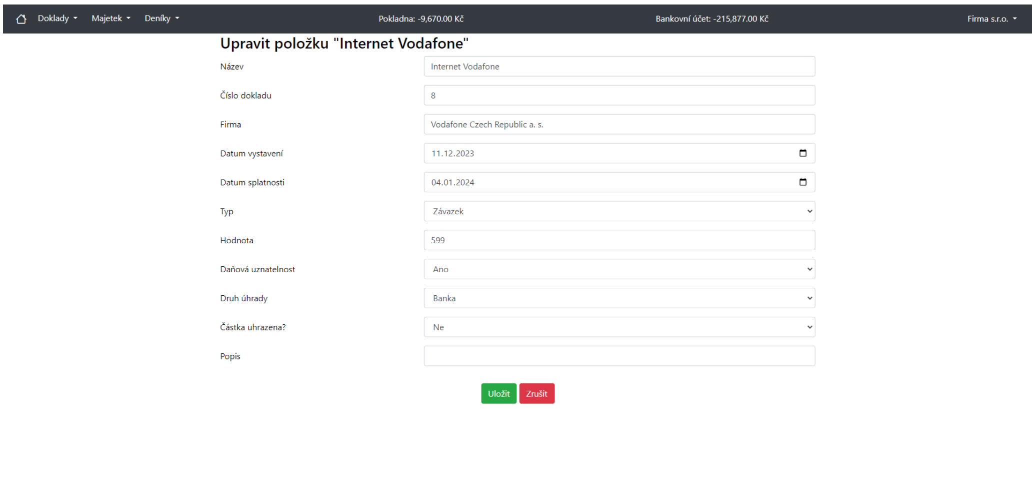
(i) A pop-up window with information about the asset’s depreciation



Upravit položku "NZ UNI Elektrocentrála"

Název	NZ UNI Elektrocentrála
Doklad	5
Datum uhrazení	11.10.2021
Příjem nebo výdaj	Výdaj
Částka	4700
Daňová uznatelnost	Ano
Druh úhrady	Banka
Popis	

(j) Edit page for income and expense



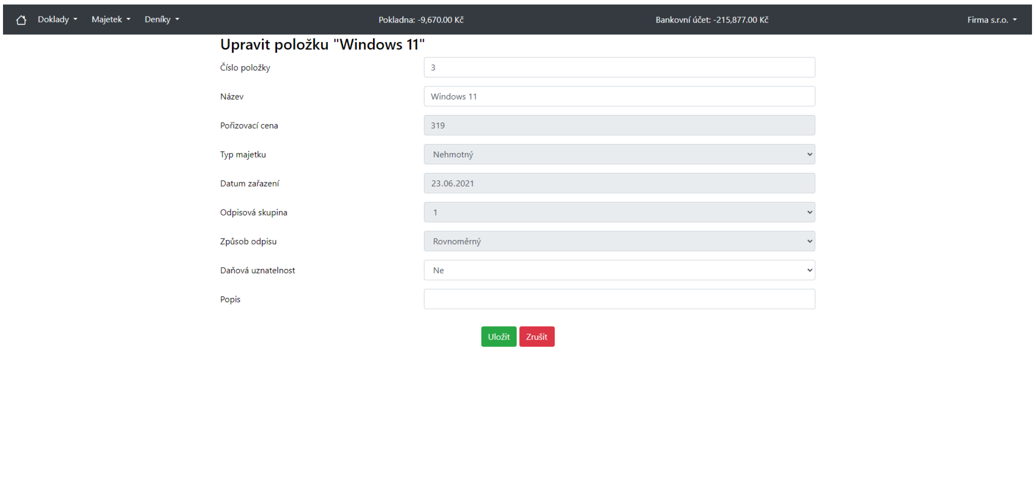
Pokladna: -9,670.00 Kč

Bankovní účet: -215,877.00 Kč

Upravit položku "Internet Vodafone"

Název	Internet Vodafone
Číslo dokladu	8
Firma	Vodafone Czech Republic a. s.
Datum vystavení	11.12.2023
Datum splatnosti	04.01.2024
Typ	Závazek
Hodnota	599
Daňová uznatelnost	Ano
Druh úhrady	Banka
Částka uhrazena?	Ne
Popis	

(k) Edit page for demand and debt



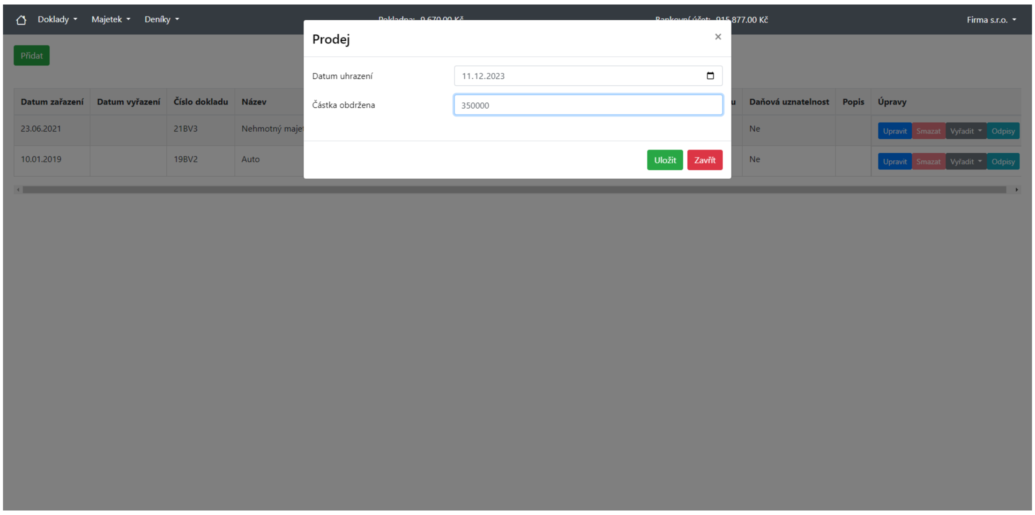
Pokladna: -9,670.00 Kč

Bankovní účet: -215,877.00 Kč

Upravit položku "Windows 11"

Číslo položky	3
Název	Windows 11
Pořizovací cena	319
Typ majetku	Nehmotný
Datum zařazení	23.06.2021
Odpisová skupina	1
Způsob odpisu	Rovnoměrný
Daňová uznatelnost	Ne
Popis	

(l) Edit page for demand and debt

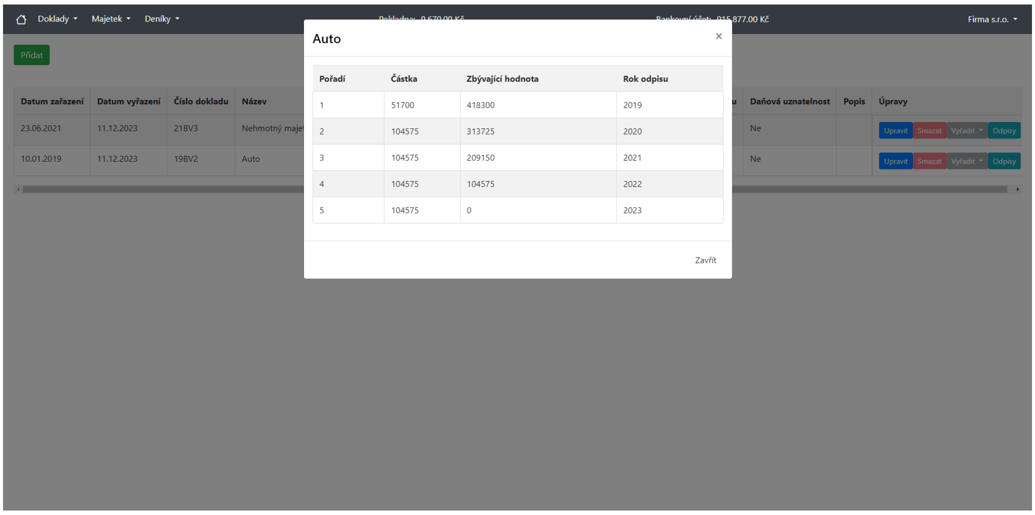


Prodej

Datum uhrazení: 11.12.2023

Částka obdržena: 350000

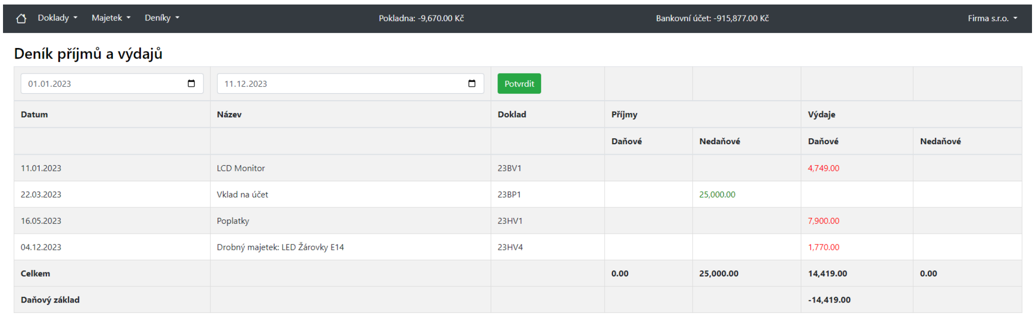
(m) A pop-up window for eliminating the asset in a case of sale



Auto

Pořadí	Částka	Zbývající hodnota	Rok odpisu
1	51700	418300	2019
2	104575	313725	2020
3	104575	209150	2021
4	104575	104575	2022
5	104575	0	2023

(n) Adjusted calculation with the elimination in the last row



Pokladna: -9,670.00 Kč

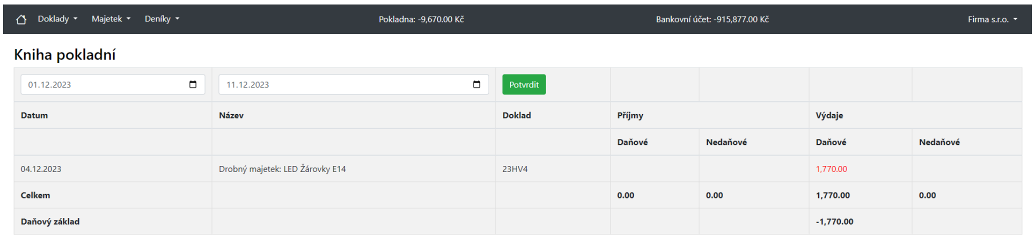
Bankovní účet: -915,877.00 Kč

Deník příjmů a výdajů

01.01.2023 – 11.12.2023

Datum	Název	Doklad	Příjmy		Výdaje	
			Daňové	Nedaňové	Daňové	Nedaňové
11.01.2023	LCD Monitor	23BV1			4,749.00	
22.03.2023	Vklad na účet	23BP1		25,000.00		
16.05.2023	Poplatky	23HV1			7,900.00	
04.12.2023	Drobný majetek: LED Žárovky E14	23HV4			1,770.00	
Celkem			**0.00**	**25,000.00**	**14,419.00**	**0.00**
Daňový základ					**-14,419.00**	

(o) Income and expense journal page



Pokladna: -9,670.00 Kč

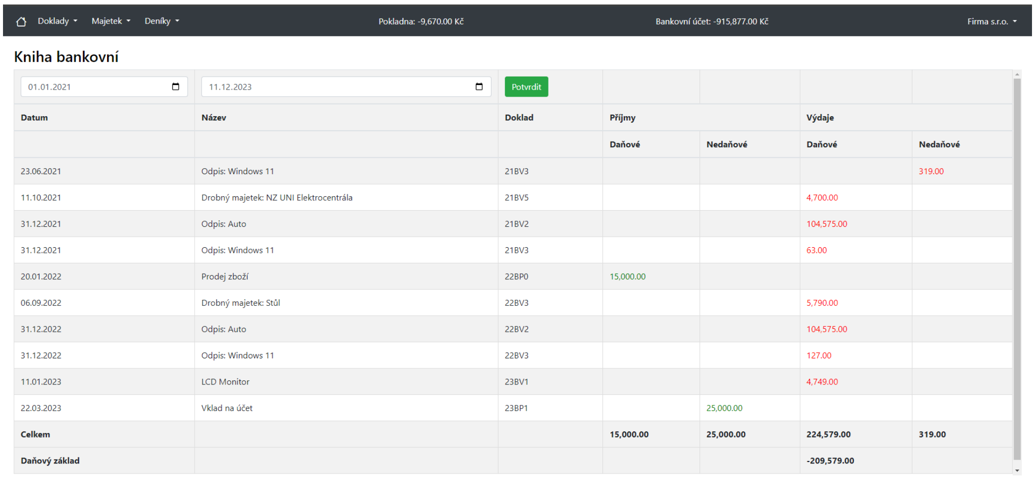
Bankovní účet: -915,877.00 Kč

Kniha pokladní

01.12.2023 – 11.12.2023

Datum	Název	Doklad	Příjmy		Výdaje	
			Daňové	Nedaňové	Daňové	Nedaňové
04.12.2023	Drobný majetek: LED Žárovky E14	23HV4			1,770.00	
Celkem			**0.00**	**0.00**	**1,770.00**	**0.00**
Daňový základ					**-1,770.00**	

(p) Cashbook page

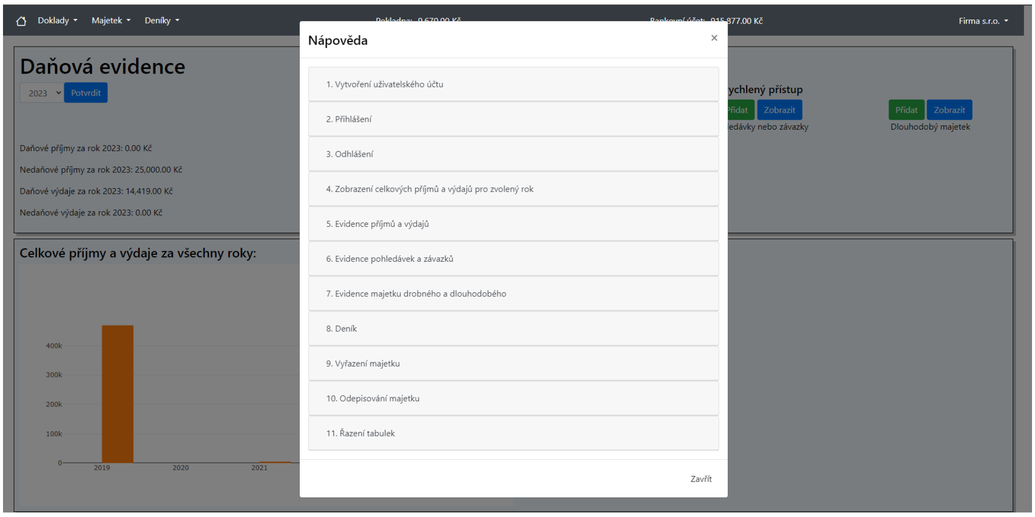


Kniha bankovní

01.01.2021 – 11.12.2023

Datum	Název	Doklad	Příjmy		Výdaje	
			Daňové	Nedaňové	Daňové	Nedaňové
23.06.2021	Odpis: Windows 11	21BV3				319.00
11.10.2021	Drobný majetek: NZ UNI Elektrocentrála	21BV5			4,700.00	
31.12.2021	Odpis: Auto	21BV2			104,575.00	
31.12.2021	Odpis: Windows 11	21BV3			63.00	
20.01.2022	Prodej zboží	22BP0	15,000.00			
06.09.2022	Drobný majetek: Stůl	22BV3			5,790.00	
31.12.2022	Odpis: Auto	22BV2			104,575.00	
31.12.2022	Odpis: Windows 11	22BV3			127.00	
11.01.2023	LCD Monitor	23BV1			4,749.00	
22.03.2023	Vklad na účet	23BP1		25,000.00		
Celkem			**15,000.00**	**25,000.00**	**224,579.00**	**319.00**
Daňový základ					**-209,579.00**	

(q) Bankbook page



Daňové příjmy za rok 2023: 0.00 Kč

Nedaňové příjmy za rok 2023: 25,000.00 Kč

Daňové výdaje za rok 2023: 14,419.00 Kč

Nedaňové výdaje za rok 2023: 0.00 Kč

Nápověda

1. Vytvoření uživatelského účtu
2. Přihlášení
3. Odhlášení
4. Zobrazení celkových příjmů a výdajů pro zvolený rok
5. Evidence příjmů a výdajů
6. Evidence pohledávek a závazků
7. Evidence majetku drobného a dlouhodobého
8. Deník
9. Vyřazení majetku
10. Odepisování majetku
11. Řazení tabulek

(r) A guide pop-up window