

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Aleksei Lavrov 154884IAPB

OPEN-SOURCE WEB APPLICATION FOR MANAGING EURO COIN COLLECTION

Bachelor's thesis

Supervisor: Gert Kanter
PhD

Tallinn 2021

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Aleksei Lavrov 154884IAPB

**AVATUD LÄHTEKOODIGA
EUROMÜNTIDE KOLLEKTSIONEERIMISE
VEEBIRAKENDUS**

Bakalaureusetöö

Juhendaja: Gert Kanter
PhD

Tallinn 2021

Author's declaration of originality

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

Author: Aleksei Lavrov

25.05.2020

Abstract

This thesis is focused on the implementation of the open-source web application for managing euro coin collection.

There are multiple applications for managing euro coin collection and trading euro coins. However, as it is described in the thesis, some applications are not open-source, contain advertisements, limited features for managing coin collection and other drawbacks.

The aim of this thesis is to create such a web application which overcomes listed drawbacks and makes it possible for a user to trade coins with other users. As a result, the mentioned application is implemented. Validation results satisfy set goals and requirements. Future development plans are described.

This thesis is written in English and is 41 pages long, including 7 chapters, 9 figures and 2 tables.

Annotatsioon

Avatud lähtekoodiga euromüntide kolleksioneerimise veebirakendus

Antud lõputöös keskendutakse avatud lähtekoodiga euromüntide kolleksioneerimise rakenduse arendamisele.

Eksisteerib palju erinevaid lahendusi euro müntide kolleksiooni haldamiseks. Selle töö käigus analüüisiti olemasolevaid lahendusi ning tuldi järeldusele, et erinevates arendatud rakendustes on puuduseid. Esiteks, mõned rakendused sisaldavad reklaame, piiratud funktsionaalsust kolleksiooni haldamiseks, mõned ei ole kättesaadavad erinevates seadmetes. Samuti olemasolevad rakendused ei ole avatud lähtekoodiga.

Eesmärgiks on arendada sellist euromüntide kolleksioneerimise rakendust, mis oleks avatud lähtekoodiga, milles on kõrvaldatud teistes rakendustes eelnevalt mainitud puudujäägid. Rakendus peab rahuldama püstitatud eesmarke ja nõudmisi. Lisaks luuakse selline funktsionaalsus, mille abil kasutajad saaksid vahetada münte teiste kasutajate vahel.

Saavutatud tulemus on valideeritud automaatsete ja potentsiaalsete kasutajatega läbiviidud intervjuu abil. Valideerimine õnnestus. Peale läbiviidud intervjuud jõuti järeldusele, et tulevikus on võimalik teha muudatusi, et nimetatud veebirakendus paremini vastaks kasutajate nõudmistele.

Töö käigus on välja arendatud selline veebirakendus, mis rahuldab püstitatud eesmarke ja nõudmisi. Lõputöö kokkuvõtte sisaldab samuti antud veebirakenduse tulevikus arendamise lisavõimalusi.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 41 leheküljel, 7 peatükki, 9 joonist, 2 tabelit.

List of abbreviations and terms

API	Application programming interface
Back end	Part of an application, for storing and manipulating data
CSS	Cascading Style Sheets
EU	European Union
Express.js	Back-end web app framework for Node.js
Front end	Part of an application, for interacting with graphical interface
HTML	HyperText Markup Language
I/O	Input/output
JavaScript	Scripting language
Node.js	Back-end JavaScript runtime environment
OS	Operating system
PostgreSQL	Relational database management system
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Table of contents

1 Introduction	10
2 Background.....	12
2.1 Existing Solutions	13
3 Analysis	16
3.1 Requirements	16
3.2 Choice of Technology	17
3.2.1 HTML.....	18
3.2.2 CSS	18
3.2.3 JavaScript	19
3.2.4 Node.js.....	20
3.2.5 PostgreSQL.....	21
3.2.6 Express.js.....	21
4 Implementation.....	23
4.1 Front-End Development	23
4.2 Back-End Development.....	26
5 Results	30
5.1 Results Validation.....	30
5.1.1 Test automation	30
5.1.2 User experience	31
6 Future Work.....	33
7 Summary.....	34
References	35
Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis	37
Appendix 2 – Application responsive layout in a device with screen size 768 x 1024 pixels.....	38
Appendix 3 – Login page	39
Appendix 4 – Application screenshot. Example	40
Appendix 5 – Survey results presented in separate charts.....	41

List of figures

Figure 1. HTML code example.	18
Figure 2. CSS code example.....	19
Figure 3. JavaScript code example.	20
Figure 4. User authentication flow diagram.	24
Figure 5. Logged in environment.	24
Figure 6. Example. Responding to coin swap offer.	25
Figure 7. Example. Swap tab. Sent swap requests.	26
Figure 8. Entity relationship diagram.	28
Figure 9. Example. Database insert query.....	29

List of tables

Table 1. Coin grading system used in NGC.....	13
Table 2. Comparison of app features.....	15

1 Introduction

The euro is the official currency of 19 European Union countries which collectively make up the euro area, also known as the eurozone [1].

As there are many different euro coins in circulation and on sale, coin collecting may become a hobby for a person. Getting various coins from circulation is easy and relatively inexpensive. The goal of collecting coins for a hobbyist is usually to make a complete collection of desired coins according to chosen theme of collection.

The design of different euro coins varies from country to country. Moreover, some countries change the design of issued euro coins from time to time. Also, various countries may issue euro coins in different denominations.

Due to the variety of issued euro coins, a coin collection may become big, so collectors may need a way of managing their collections to have an overview of collected coins.

Moreover, as it is mentioned above, the euro is the currency of multiple countries, so there may be many potential hobbyists who collect euro coins.

For the reasons described above, the author decided to create a web application for managing euro coin collection.

The aim of developing a web application is to create such application that fulfils the next main purposes:

- Application is accessible on various devices
- Application source code is open
- Application has no ad
- Possibility to add coins to collection
- Possibility to swap coins

Detailed requirements for the application are to be made after reviewing of existing solutions.

The result of implementation of the application is required to be analyzed after conducting a survey among potential users and performing test automation.

2 Background

Coin collecting is the collecting of coins or other forms of minted legal tender. Coin collecting can be differentiated from numismatics, in that the latter is the systematic study of currency, though the two disciplines are closely interlinked [2].

A person can collect coins by getting coins from circulation, for example, received a change for a purchase, bought at a bank or at a coin shop.

The motivations for collecting vary from one person to another [2]. Some collect coins on occasion when they find some interesting coin. For another it is a hobby, for example, collecting coins by getting them from circulation and without spending much money to get some specific coins to make a collection. Some people invest in coins by buying collector coins which are often made of precious metals, for example, silver or gold.

There are various themes for coin collection. For example, coin collecting by issue date, by mint marks, by series, by country or region, by nominal or by another specific theme.

As a collectible, a coin value depends on its grading. In numismatics, coins are graded by specialists or special organizations.

The table below (see Table 1) [3] represents coin grading system by NGC (Numismatic Guaranty Corporation) [3].

Abbreviation	Numeric value	Grade
MS	60–70	Uncirculated
AU	50, 53, 55, 58	About Uncirculated
XF	40, 45	Extremely Fine
VF	20, 25, 30, 35	Very Fine
F	12, 15	Fine
VG	8, 10	Very Good
G	4, 6	Good
AG	3	About Good
FA	2	Fair
PR	1	Poor

Table 1. Coin grading system used in NGC.

Coin collection can be stored in special albums, boxes, folders or even in a can. If a collection consists of many coins, it can be useful to add coins to tables or apps to have overview of collectibles.

2.1 Existing Solutions

While searching for various existing solutions for managing euro coin collection, the main goal is to find such euro coin collecting apps that are open-source, free and accessible on numerous devices.

Although there are plenty of such applications on Google Play available for devices with Android operating system, it is difficult to find an application that is completely free and contains various collection management solutions. The author chose EURik application for this comparison because it has more than 10 000 downloads [4].

However, no applications on App Store for storing euro coin collection for devices with iOS are chosen for this comparison because available applications are not free or are with limited functionality.

To compare various existing solutions, two more applications were chosen for the comparison. First, *Coincollector.eu* is chosen based on the result of searching on Google with „collect euro coins“ keywords. Second, *Euro-muenze.tv* is chosen for comparison because it contains a rich euro coin catalogue.

The table below (see Table 2) represents a comparison of coin collecting features in the applications mentioned before.

Application	EURik	Coincollector.eu	Euro-muenzen.tv
Feature			
Availability			
Browser	-	+	+
Android (OP system)	+ ¹	+	+
iOS (OP system)	-	+	+
Language			
English	+	+	-
Estonian	-	+	-
German	+	-	+
User account			
Registration	-	+	+
Google account	-	+	-
Facebook account	-	+/- ²	-
Sorting coins			
By countries	+	+	+
By years	+	+	+/- ³
By nominals	+/- ⁴	+/- ⁴	+/- ³
By commemoratives	+	+	+
By precious metals	-	-	+/- ³
By missing coins	-	+	-
Adding coin features			
Colorized coins	-	-	+/- ⁵
Coin sets	-	-	+
Coin types	+	+/- ⁶	+
Coin grades	+	+	-
Setting coin value	+	-	+
Managing coin collection			
Share collection link	-	+	-
Collected coins overview	+	+	-

Application	EURik	Coincollector.eu	Euro-muenzen.tv
Feature			
Collected coins statistics	+	+	-
Export collection	+	+	-
Added coins timeline	-	+	-
Favourite coins	-	+	-
Wanted coins	-	-	-
Other			
Coin swaps	-	-	-
Open-source app	-	-	-
Ad-free	-	+	-
<p>¹ Available only as app for Android OP system</p> <p>² Can connect a Facebook account after registration</p> <p>³ Only when country / coin type is chosen</p> <p>⁴ Other nominals unavailable, e.g., 3 euro, etc.</p> <p>⁵ Officially issued colorized coins</p> <p>⁶ Only German mintage coin types available</p>			




 Implemented feature
 Partly implemented feature
 Not implemented feature

Table 2. Comparison of app features.

3 Analysis

The given chapter represents an analysis of requirements for the application and the technologies which are chosen to develop the application.

3.1 Requirements

Before implementing the application, it is important to define requirements to understand what and how the application should do some actions. The requirements are split up to functional and non-functional requirements.

Functional requirements define the facilities that are available in the system. Functional requirements for the application mentioned before are presented in the following list:

- Authentication of the user when user uses application
- While logged in, user can change name, email, or password
- On the main page, coins should be displayed by coin categories
- Application should have the opportunity to view coins by specific country
- Application should have the opportunity to view coins by specific denomination
- Application should have the opportunity to view coins by commemorative issues
- Application should have the opportunity to view coins by precious metals
- Each coin with detailed parameters can be viewed in separate page
- User can add a coin to collection with specific coin parameters
- Added coin must be differed from other coins when displayed in some coin category
- User can change added coin parameters
- User can delete added coin
- User can add wanted coin with specific parameters to wanted coins list
- User can change wanted coin parameters
- User can delete wanted coin
- The number of added coins should be displayed in the progress bar
- The overview of added coins should be available in the Statistics tab

- User can respond to another user coin swap request by composing and submitting desired swap request
- User can create swap offers depending on another user's wanted coins list
- User can respond to another user coin swap offer by composing (depending on another user wanted coins list) and submitting desired swap request
- Offered coins in the swap request can be changed
- Swap request can be deleted
- Application should not include ads

Non-functional requirements define the quality features of the system. Non-functional requirements for the application are presented in the following list:

- Application should be accessible through browser on various devices with different OS and browsers
- Application should have responsive layout on smaller screens
- Application should be available in English language
- Several users may use application without impacting the performance
- Users, except the administrator, should not be allowed to add initial coins to the database
- User data cannot be modified in the application while user is logged out
- User data cannot be accessed in the application while user is logged out
- Application should have open-source code

3.2 Choice of Technology

Several technologies are chosen to implement an application which meets the requirements described in the previous subchapter (see chapter 3.1). The implementation process is split up to front-end and back-end development. The app must have a user interface which is implemented in the front-end of the app. The user interface is connected with server and database, which is implemented in the back end of the app.

Technologies which will be used in the front-end development are HTML, JavaScript and CSS. Technologies which will be used in the back-end development are Node.js, PostgreSQL and Express.js. The choice and overview of these technologies is described in the next subchapters (see chapter 3.2.1 HTML – 3.2.6 Express.js).

3.2.1 HTML

HTML (*HyperText Markup Language*) is the most basic building block of the Web. It defines the meaning and structure of web content [5].

The first version of HTML was written by Tim Berners-Lee in 1993. Since then, there have been many different versions of HTML. The most widely used version throughout the 2000's was HTML 4.01, which became an official standard in December 1999 [6].

The code below (see Figure 1) represents a simple HTML document source code.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Document title</title>
</head>
<body>
  <h1>Heading</h1>
  <span>Some text</span>
</body>
</html>
```

Figure 1. HTML code example.

HTML5, the latest version of HTML, is used in the mentioned web application development because it is the most common technology which is recognized by every web browser.

3.2.2 CSS

CSS (*Cascading Style Sheets*) is a stylesheet language used to describe the presentation of a document written in HTML or XML. CSS describes how elements should be rendered on screen, on paper, in speech, or on other media [7].

CSS is among the core languages of the open web and is standardized across Web browsers according to W3C specifications [7].

The code below (see Figure 2) represents a simple CSS code example.

```
p {  
    color: blue;  
    background-color: grey;  
    font-size: 16px;  
    font-weight: bold;  
}
```

Figure 2. CSS code example.

The latest CSS release, CSS 2.1 : Level 2 Revision 1, is being used while developing mentioned application.

3.2.3 JavaScript

JavaScript (*JS*) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. It is most well-known as the scripting language for Web pages. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g., functional programming) styles [8].

The standards for JavaScript are the ECMAScript Language Specification (ECMA-262) and the ECMAScript Internationalization API specification (ECMA-402) [8].

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM) [9].

JavaScript is the dominant client-side scripting language of the Web, with 97% of websites using it for this purpose. Scripts are embedded in or included from HTML documents and interact with the DOM. All major web browsers have a built-in JavaScript engine that executes the code on the user's device [9].

The code example below (see Figure 3) represents an extended example from Figure 1.

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Document title</title>
</head>
<body>
  <p>Page title</p>
  <span class="example">Some text...</span>
  <button type="button" onclick="changeText();" >
    Change text</button>
  <script>
    function changeText() {
      document.querySelector(".example").innerHTML = "New text...";
    }
  </script>
</body>
</html>

```

Figure 3. JavaScript code example.

The latest release of JavaScript, ECMAScript 2020, is being used to develop mentioned application.

3.2.4 Node.js

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser [10].

Node.js can generate dynamic page content, create, modify, and delete files on the server, collect form data, add, modify, and delete data in a database [11].

Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many I/O operations, as well as for real-time Web applications [10].

Node.js was written initially by Ryan Dahl in 2009 [10].

Mentioned technology it is widely used among many companies such as Microsoft, Netflix, PayPal, and others [10].

For the reasons given above, it is decided to use Node.js in the application back-end development.

3.2.5 PostgreSQL

PostgreSQL is an enterprise-class open-source database management system. It supports both SQL and JSON for relational and non-relational queries for extensibility and SQL compliance [12].

PostgreSQL (initially called Postgres) was created by a computer science professor Michael Stonebraker and his team in 1986. Today it has become one of the most popular open-source databases [12].

Some key features [12] and advantages [12] of PostgreSQL:

- Compatible with various platforms using all major languages and middleware
- Mature Server-Side Programming Functionality
- Support for JSON data
- Can run dynamic websites and web apps as a LAMP stack option
- Easy to use

It is widely used by some major organizations such as Apple, IMDB and Instagram [13].

For the reasons given above, it is decided to use PostgreSQL in the back-end development to create and manage database for the application.

3.2.6 Express.js

Express.js, or simply Express, is a back-end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs [14].

This list [15] below represents some features of Express.js:

- Write handlers for requests with different HTTP verbs at different URL paths (routes)
- Integrate with "view" rendering engines to generate responses by inserting data into templates
- Set common web application settings like the port to use for connecting, and the location of templates that are used for rendering the response

- Add additional request processing "middleware" at any point within the request handling pipeline

While Express itself is minimalist, developers have created compatible middleware packages to address almost any web development problem [15].

Express.js was founded by TJ Holowaychuk. The first release, according to Express.js's GitHub repository, was on the 22nd of May, 2010 [14].

Express.js is used by some organizations such as Uber [16], IBM [16], Yandex [16] and others [16].

For the reasons give above, the author decided to use Express.js framework in the application back-end development.

4 Implementation

As it was mentioned in chapter 3.1, the application for managing euro coin collection should be accessible on various devices. For this reason, it is decided to create a web application because a web application is supposed to be accessible on various devices through browser connected to the Internet. So, the application data to be stored in the database and can be accessed by sending a request from the user interface to the server.

One of the main goals is that the web application should be open-source. So, the author decided to host the source code in public a GitHub repository [17]. For more simplicity, it is decided to give the web application a name *Crispyeuro*.

The given chapter describes the implementation process and features of the application for managing euro coin collection and includes to two subsections. The first section describes the front-end development, and the second section describes the back-end development.

4.1 Front-End Development

Front-end development describes the implementation of the user interface and interaction with it on the client side.

The first step is to develop user interface by creating HTML documents assisted with CSS to style them and with JavaScript to make HTML documents dynamic so that user could interact with them.

When user opens web application in a browser it is required to log in or sign up.

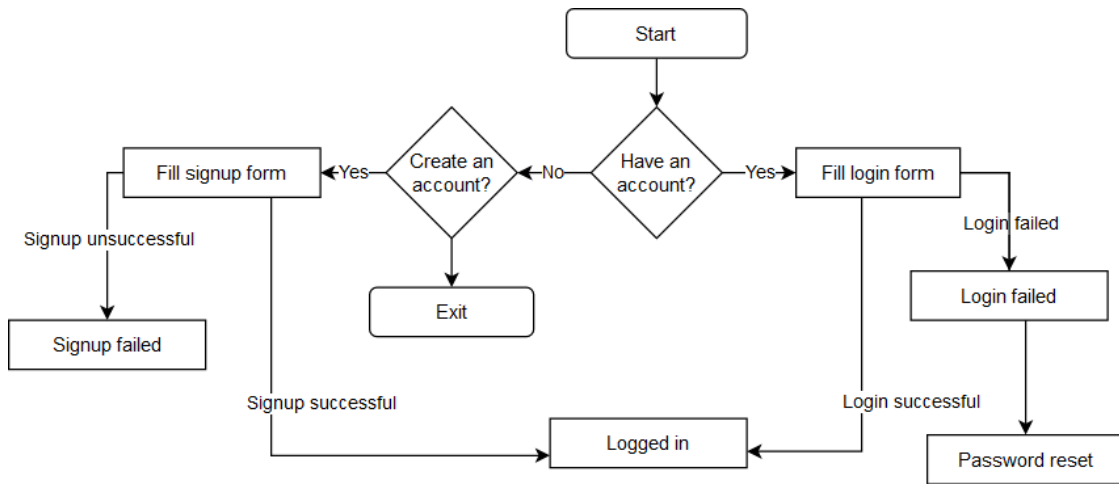


Figure 4. User authentication flow diagram.

The diagram above (see Figure 4) describes user authentication process. When login is successful, user is redirected to logged in environment (see Figure 5).



Figure 5. Logged in environment.

To view a coin, user should open desired coin category from the main page or from the side menu and choose a country or a year. Then, user can choose and open desired coin. When coin is opened, user can view detailed information about the chosen coin.

To add coin to the collection, user may click the button *Add coin* and submit desired information about the coin, for example, grade, amount of such coins, design etc. To create a swap offer, users may check the *Swap availability* checkbox near the added coin to make the coin visible to other users in the *coin* page in *Swap availability* section.

To add a coin to the wanted coins list, user may click the button *Want this coin* in the *coin* page and, if desired, create a coin swap request with desired coin details. So, this swap request becomes visible to other users in the *coin* page in *Swap availability* section.

To respond to the coin swap request or offer, user should click on the username in the *coin* page the *Swap availability* section in the end of the opened *coin* and then choose one or more coins to offer and/or desired coin or coins.

Figure 6 below represents the process responding to the coin swap request. The first part from the left (a) represents details of the offered coin, and the second part (b) represents other user's wanted coins list and user's list of coins which can be offered for swap.

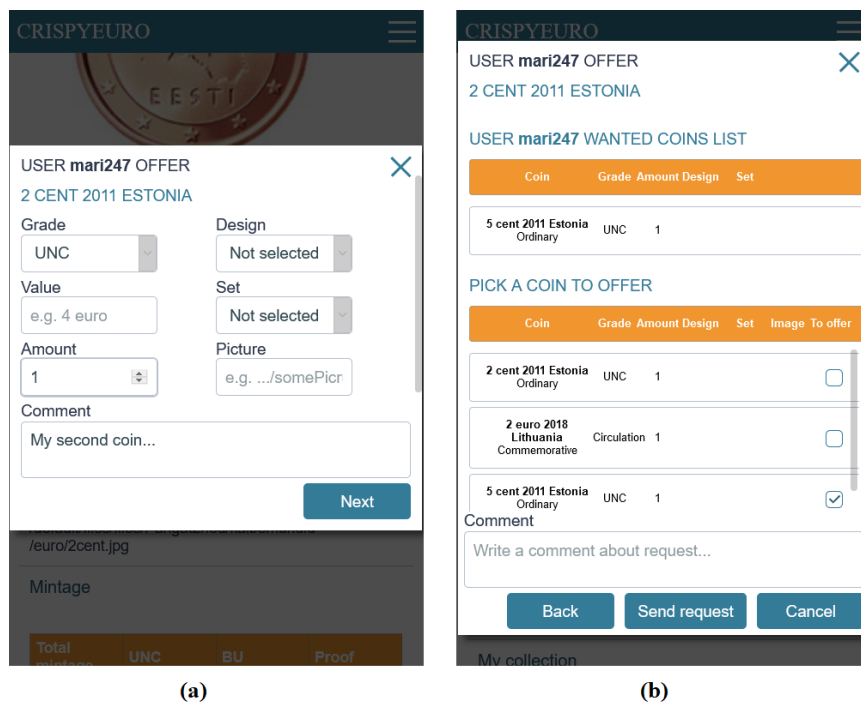


Figure 6. Example. Responding to coin swap offer.

To manage sent or received coin swap request, user should click on the *Swap* button in the top navigation bar to view coin requests. In the *Swap* tab (see Figure 7) user can view offered or requested coins' details, create a conversation with request receiver, and cancel or dismiss the coin swap request.

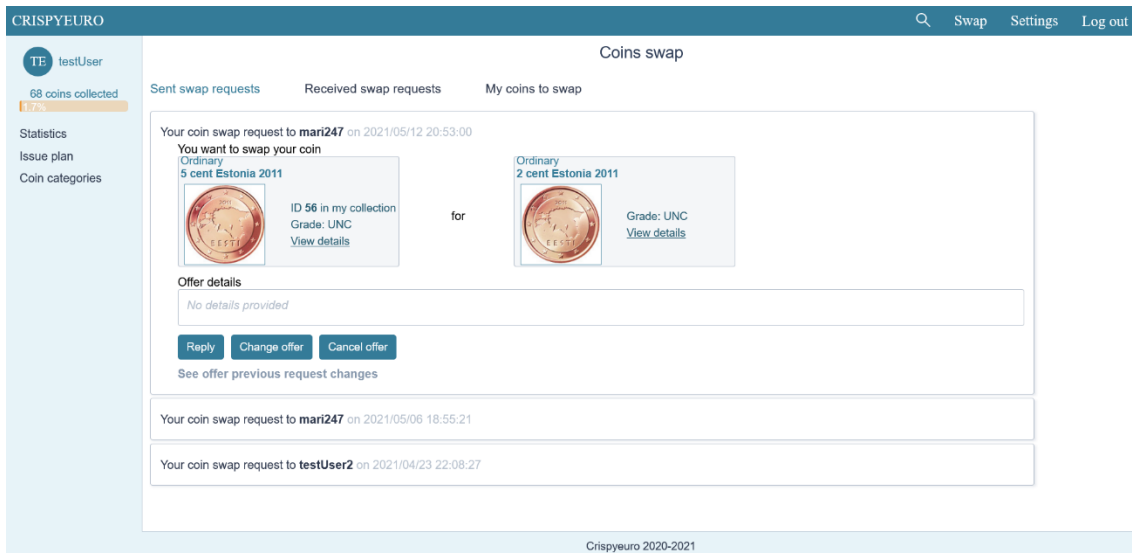


Figure 7. Example. Swap tab. Sent swap requests.

User can change offered coins in the *Sent swap requests* tab in desired swap request and these changes are visible afterwards in the *See previous changes* subtab of the swap request.

If coins are physically swapped, the swap request receiver may push the button *Swapped* in the *Received swap requests* subtab in desired swap request to make the app to perform the swapping process.

There are also implemented some other features. To change user data, user may click the button *Settings* in the top navigation menu to fill in and submit the form with changes. To see collection statistics in details, user may push the button *Statistics* in the side navigation menu to see collection details.

To make the web application responsive on smaller screens, the top and side navigation menus' styles were changed with *Flexbox layout* CSS modules to be responsive depending on the user screen size. Examples of responsive user interface can be seen in the Appendix section.

4.2 Back-End Development

Back-end development describes the implementation of the database and server. Database should store the information about users, coins, and swap requests. Server should interact with user interface and the database.

The first step is to set up database and server environment. To set up database, PostgreSQL should be installed and run. Then, the database can be created, using command “CREATE DATABASE crispyeuro;” in the SQL Shell command line.

The second step is to set up server environment. Server environment is to be set up after installing Node.js. To develop the application, there are also used some Node.js modules.

First, it is required to install node-postgres. Node-postgres is a collection of Node.js modules for interfacing with PostgreSQL database. [18] It can be installed from the command line with the next command: “npm install pg”. Second, Express.js and cookie-parser should also be installed.

When the environment is set up, the application can run from directory “crispyeuro-server/” with the command “npm run start”. Then, the application can be accessed in a browser by entering an address “http://127.0.0.1:8080/”. When address is entered, the file “server.sh” is run. Mentioned file contains commands to get access to database and the command to run the server.

To store the data about users, coins, and swap requests it is needed to create database tables.

The diagram below (Figure 8) shows created PostgreSQL database tables and relationships.

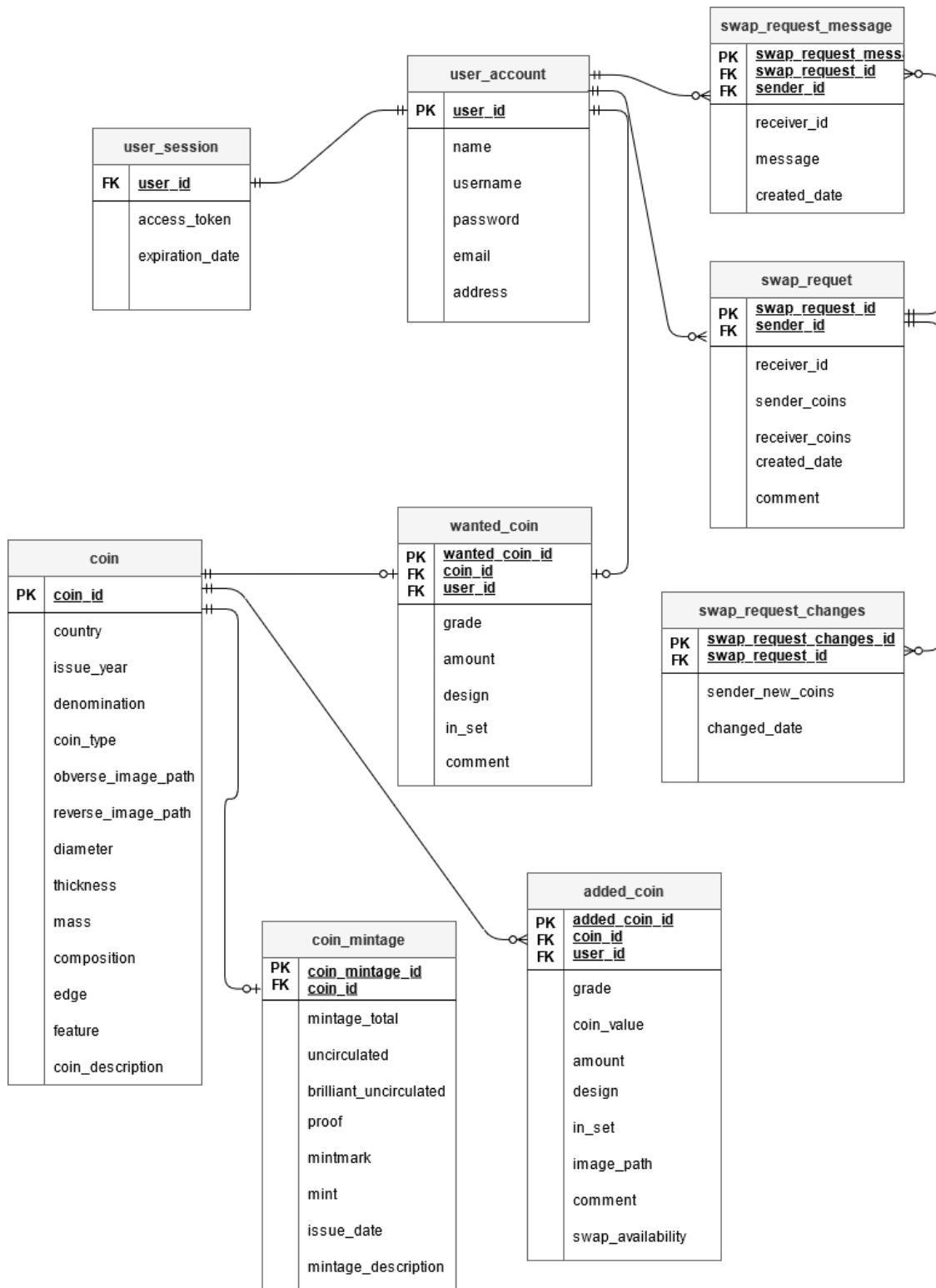


Figure 8. Entity relationship diagram.

If new user is added to the database, it is needed to create user session so that user could log in. User session data is stored in the database. For this purpose, the table “user_session” is created. When user enters username and password in the user interface to log in, sent data is validated on the client-side and on the server-side. When validation

is successful, new user session is created when unique “access_key” is generated, and user session expiration date is created, and both are added to database table “user_session”. User session is deleted from the database after 7 days if user did not log out. If user logs out, user session expires immediately.

To make it possible for user to view and add new coins to collection, it is needed to add coins to the database. Coins and coin mintages are added to database by admin user. In the “coin” table, it is required to insert at least coin country, issue year and denomination.

The code below (see Figure 9) shows the insertion of the coin data into “coin” database table.

```
INSERT INTO coin(country, issue_year, denomination, coin_type,
diameter, thickness, mass, composition, edge)

VALUES ('Estonia', 2011, '0.01', 'ordinary', 16.25, 1.67, 2.3,
'Copper-covered steel', 'Smooth');
```

Figure 9. Example. Database insert query.

Coin swap requests are stored in the “swap_request” table. Users may have any coins in any amount in a swap request. It is possible, for example, to create a swap request with no offered coin, if such coin does not exist in the app database, but user has the coin to offer. So, in this case, swap offer details may be mentioned in the *Comment* field.

Coin swap can be initialized in the application by swap request receiver. When such database query is sent to server, PostgreSQL executes function “swap_change_added_coins_owner” and changes users’ “user_id” vice versa, empties *comment* values and makes coins not available for swap by default.

5 Results

As a result, the application development process is finished. The application meets the main goals which were set in the beginning (see chapter 1).

However, to make sure that implemented web application satisfies set requirements (see chapter 3.1), it is important validate achieved results.

5.1 Results Validation

Results are validated in two steps. First, automation tests are executed to make sure that application works without mistakes. Second, feedback from potential users should be received to know how users are satisfied with the application usage experience.

5.1.1 Test automation

To perform test automation, the author decided to use several tools.

First, Selenium. It is an open-source automation testing tool which is used for automating tests carried out on different web-browsers. Some of the most important advantages of Selenium are open-source availability, multi-browser support, ease of implementation, parallel test execution [19].

Second, AVA. It is a test runner for Node.js. This tool has some useful advantages. It allows to run tests concurrently. It boasts a simple syntax, “no implicit globals,” magic assertions, promise and async function support, observable support, and enhanced assertion messages [20].

Third, NYC. It is a npm package for getting statistics about the test coverage. This tool is needed to get the information about how much of the code is covered with tests. After execution of this tool, NYC generates the report which shows how many lines of the code are covered with tests [21].

The tests, which were created to test the application, can be executed from the command line in “crispyeuro-client” directory with the “npm t” command. When the tests are run, the process of testing through the application graphical interface can be seen in the browser. After test automation finished, there will be generated NYC test coverage report which should show how many lines of the code are covered with tests.

As a result, test automation is successful, and this means that the application works as required.

5.1.2 User experience

The second part is to validate user experience. It is important to get feedback from potential users to know how users are satisfied with the application and get ideas to improve the functionality and features.

To get feedback from potential users, the author decided to conduct a survey. The survey is created by using Google Forms and consists of 6 questions.

As a result, 3 potential users were questioned, and results received. Results are presented in separate charts in Appendix 5.

Answers to two first questions “Do you collect euro coins?” and “May a web application be a good option for managing euro coin collection?” are received to make sure that user is interested in collecting euro coins and managing coin collection in a web application.

The third question “Please create an account for CRISPYEURO app and add 1 coin to collection. Please describe your experience” is asked to get feedback about viewing and managing coin collection. Received answers are “Easy” and “Good” which means that feedback about mentioned functionality is positive.

The fourth question “Please create a coin swap request. Please describe your experience” is asked to get feedback about the functionality of trading coins. Received answers are “Easy”, “Good, but swapping coins is a little bit complicated” and “Good”. To make a conclusion from this feedback, mentioned functionality is mostly good, but it is needed to simplify the functionality of trading coins and find a way to rearrange coin swap options to make it more intuitive.

The fifth question “How are you satisfied with using CRISPYEURO web application?” (on the scale of 0 to 10) is asked to get overall feedback about the application. Received answers are “8”, “9” and “10”. This feedback means that the satisfaction with the application is good. However, according to previous answers, some improvements are needed to be made.

The last question is “Do you have any ideas for improving CRISPYEURO web application?”. Received answers for this question are “No”, “Make swapping coins easier” and “Watching coin tables is less comfortable on mobile version. Maybe make smaller tables or another way or viewing coins in a coin category”. In addition to the suggestion of making the functionality of trading coins simpler, it is suggested to represent big coin tables in another way to make it easier to view coins on smaller screens.

To draw a conclusion from the mentioned survey results, the feedback about the application is good, but some changes are to be made in the future. First, it is needed to make the functionality of trading coins less complicated, make this functionality more informative and find an opportunity to rearrange coin trading options to make it more intuitive for users. Second, it would be good to find another way to present big coin tables in some coin categories, for example, in *Denominations* coin category.

To sum up achieved validation results, the application satisfies set goals and meets set requirements.

6 Future Work

The main goal of the future work is to put the application on a web hosting platform so that users could manage their coin collection and trade coins on the server.

In addition to the implemented application, there are some possibilities to continue application development in the future to satisfy the needs of the users.

First, implementing functionality to export coin collection to device. This possibility may be needed to store coin collection in a separate place to share collection data with others, or if the Internet or the application is not available for some reason.

Second, implementing user roles and give chosen users the right to insert new coins to the database or edit desired coin data. This functionality could make the application autonomous.

Third, implementing the option to search for a coin by uploading a coin picture. This functionality might be useful if the user does not know which coin to search for. The coin uploaded by the user may be found by comparing coin picture with the data in the application database.

7 Summary

The aim of developing a web application was to create such application that should fulfil the next main purposes:

- Application is accessible on various devices
- Application source code is open
- Application has no ad
- Possibility to add coins to collection
- Possibility to swap coins

As a result, implemented application satisfies mentioned purposes and requirements which were listed in previous chapters.

This web application and is ready to be set up on a web hosting platform to be used by potential users.

The success of effective coin trading relies on having as many users on the platform as possible, so in that sense it would be best if there would only be one instance of the service running but development of new features could be crowdsourced since it is open source.

The validation of achieved results was successful. According to the user experience survey results, the feedback about the usability of the application is mostly good, but some changes needed to be made.

References

- [1] “The euro as the official currency of the euro area,” in *Official website of the European Union*. [Online]. Available: https://europa.eu/european-union/about-eu/euro/euro-official-currency-euro-area_en. Accessed on: May 24, 2021.
- [2] “Coin collection,” in *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Coin_collecting. Accessed on: May 24, 2021.
- [3] “The NGC coin grading system,” in *Numismatic Guaranty Corporation*. [Online]. Available: <https://www.ngccoin.com/coin-grading/grading-process/ngc-grading-process.aspx>. Accessed on: May 24, 2021.
- [4] SOM & SOFT, *EURik: Euro coins*, ver 1.9.8.5. Vitoria-Gasteiz, Spain: SOM & SOFT. [Computer software]. Available: <https://play.google.com/store/apps/details?id=com.osumsky.eurik>. Accessed on: May 24, 2021.
- [5] “HTML: HyperText Markup Language,” in *MDN Web Docs*. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTML>. Accessed on: May 24, 2021.
- [6] “A Brief History of HTML,” in *WebD2*. [Online]. Available: https://www.washington.edu/accesscomputing/webd2/student/unit1/module3/html_history.html. Accessed on: May 24, 2021.
- [7] “CSS: Cascading Style Sheets,” in *MDN Web Docs*. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/CSS>. Accessed on: May 24, 2021.
- [8] “JavaScript,” in *MDN Web Docs*. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. Accessed on: May 24, 2021.
- [9] “JavaScript,” in *Wikipedia*. [Online]. Available: <https://en.wikipedia.org/wiki/JavaScript>. Accessed on: May 24, 2021.
- [10] “Node.js,” in *Wikipedia*. [Online]. Available: <https://en.wikipedia.org/wiki/Node.js>. Accessed on: May 24, 2021.
- [11] “Node.js Introduction,” in *W3Schools*. [Online]. Available: https://www.w3schools.com/nodejs/nodejs_intro.asp. Accessed on: May 24, 2021.
- [12] “What is PostgreSQL? Introduction, Advantages & Disadvantages,” in *W3Schools*. [Online]. Available: <https://www.guru99.com/introduction-postgresql.html>. Accessed on: May 24, 2021.

- [13] Jakub Romanowski, “Which Major Companies Use PostgreSQL? What Do They Use It for?,” May 19, 2020. [Web log post]. Available: <https://learnsql.com/blog/companies-that-use-postgresql-in-business/>. Accessed on: May 24, 2021.
- [14] “Express.js,” in *Wikipedia*. [Online]. Available: <https://en.wikipedia.org/wiki/Express.js>. Accessed on: May 24, 2021.
- [15] “Express/Node introduction,” in *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction. Accessed on: May 24, 2021.
- [16] “Companies using Express in production,” in *Express*. [Online]. Available: <https://expressjs.com/en/resources/companies-using-express.html>. Accessed on: May 24, 2021.
- [17] Crispyeuro, *Crispyeuro*. [Computer software]. Available: <https://github.com/crispyeuro>. Accessed on: May 24, 2021.
- [18] *Node-postgres*. [Online]. Available: <https://node-postgres.com/>. Accessed on: May 24, 2021.
- [19] Jaswant Kaur, “11 Reasons To Use Selenium for Automation Testing,” Jan 28, 2019. [Web log post]. Available: <https://dzone.com/articles/11-reasons-why-go-for-automation-testing-using-sel>. Accessed on: May 24, 2021.
- [20] Testim, “AVA Testing Tutorial: A Guide to Lightweight Testing, ” Jan 7, 2019. [Web log post]. Available: <https://www.testim.io/blog/ava-testing-tutorial-a-guide-to-lightweight-testing/>. Accessed on: May 24, 2021.
- [21] Dany Paredes, “Code coverage in 2 minutes with NYC, ” Aug 13, 2020. [Web log post]. Available: <https://dev.to/danywalls/code-coverage-in-2-minutes-with-nyc-130m>. Accessed on: May 24, 2021.

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis¹

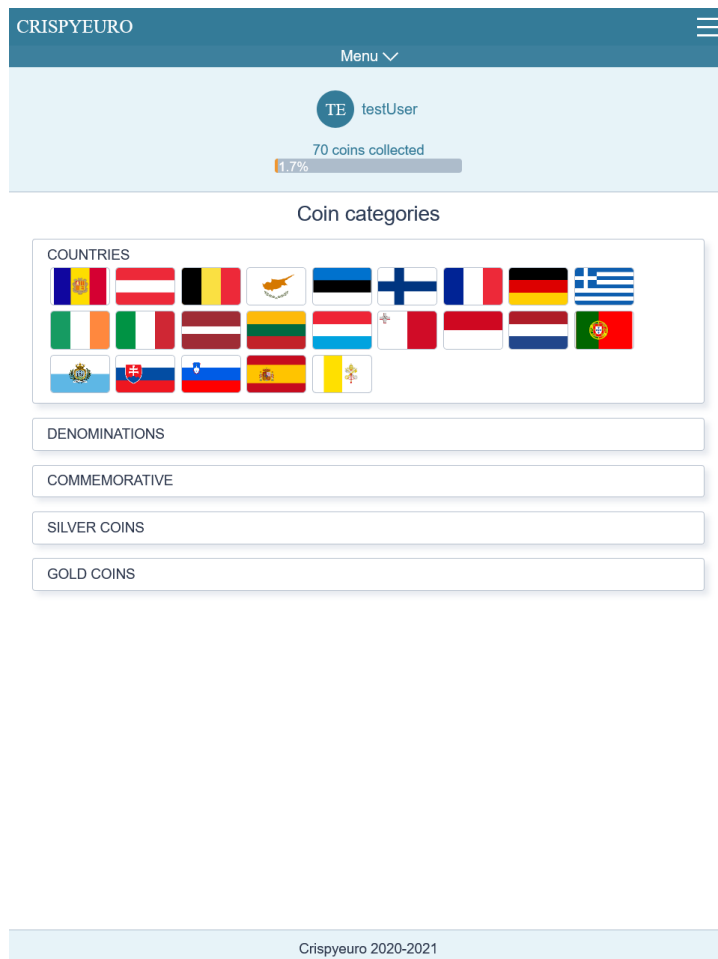
I Aleksei Lavrov

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Open-Source Web Application for Managing Euro Coin Collection" , supervised by Gert Kanter
 - 1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
 - 1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

25.05.2020

¹ The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – Application responsive layout in a device with screen size 768 x 1024 pixels



Appendix 3 – Login page

Start your euro coin collection!

Please LOG IN

Username
Enter Username

Password
Enter Password

LOG IN

[Forgot password?](#)

or

[SIGN UP](#)

CRISPYEURO
Welcome

Crispyeuro 2020-2021

Appendix 4 – Application screenshot. Example

When specific coin category is opened, user can see coins presented in separate tables. Coins, which are added to collection, are in green colour.

The screenshot shows the 'CRISPYEURO' application interface. The top navigation bar includes a search icon, 'Swap', 'Settings', and 'Log out'. The user profile 'testUser' is shown with '70 coins collected' and '1.7%' progress. The main content area is titled 'Coins of Lithuania' and includes a 'Go back' button. Below the title is a 'Description' section with the Lithuanian flag and the text 'Lithuania issued 46 ordinary coins and 12 commemorative coins.' The 'Coins' section is divided into two tables:

- Ordinary

Year	1 cent	2 cent	5 cent	10 cent	20 cent	50 cent	1 euro	2 euro
2015	View	View	View	View	View	View	View	View
2016	View							
2017	View	View		View	View			View
2018	View	View	View	View	View	View	View	View
2019	View	View	View	View	View	View	View	View
2020	View	View	View	View	View	View	View	View
2021	View	View	View	View	View	View	View	View

- Commamorative 2 euro

Order	Year	Feature	Mintage
1	2015	Lithuanian Language	No data
2	2015	Common issue. 30th anniversary of the Flag of Europe	No data
3	2016	Baltic Culture	No data
4	2017	Vilnius City of Culture	No data
5	2018	100 years since Independence. Common commemorative coin with Estonia and Latvia	No data

Appendix 5 – Survey results presented in separate charts

