Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Graphics and Interaction
Field of study: Human Computer Interaction

Content sharing in the EduARd system

MASTER'S THESIS

Author: Michaela Vlčková
Supervisor: Ing. David Sedláček, Ph.D.
Year: 2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Vlčková** Jméno: **Michaela** Osobní číslo: **407096**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Specializace: **Interakce člověka s počítačem**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Sdílení obsahu v systému EduARd

Název diplomové práce anglicky:

Content sharing in the EduARd system

Pokyny pro vypracování:

1) Seznamte se s aktuálním stavem projektu EduARd (rozšířená realita pro školy) [3, 4, 5].
2) Navrhněte a implementujte rozšíření stávajícího systému pro tvorbu učebnic o správu a sdílení 2D a 3D datového obsahu napříč portálem (tzv. tržiště obsahu).
3) Navrhněte a implementujte postup optimalizace přenášeného datového obsahu mezi aplikacemi pracujícími s portálem (editor, mobilní klientské aplikace). Spolupracujte při návrhu s kolegy řešícími další části systému (mobilní aplikace pro prohlížení obsahu).
4) Řešení průběžně testujte unit testy. Funkcionalitu ověřte vytvořením vhodného počtu kopií a úprav již existujících učebnic, jejich výběr konkretizujte s vedoucím práce. Ověřte nové učebnice na klientském zařízení. Při návrhu a realizaci uživatelského rozhraní tržiště postupujte dle metodiky UCD [2].

Seznam doporučené literatury:

[1] Fanie Reynders, Modern API Design with ASP.NET Core 2, Apress 2018
[2] T. Lowdermilk, User-Centered Design, O'Reilly Media, 2013
[3] http://eduard.fel.cvut.cz
[4] Jan Skála, Systém pro správu uživatelů projektu EduARd, BP FEL, 2019.
[5] Anastasia Surikova, Webový klient pro správu mobilních výukových úloh systému EduARd, BP FEL, 2019.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. David Sedláček, Ph.D., katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **19.01.2021** Termín odevzdání diplomové práce: **04.01.2022**

Platnost zadání diplomové práce: **30.09.2022**

Ing. David Sedláček, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Datum převzetí zadání

Podpis studentky

Declaration

I hereby declare I have written this diploma thesis independently and quoted all the sources of information used following methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

In Prague .....

.....
Michaela Vlčková

Acknowledgements

I would like to thank Ing. David Sedláček PhD for the opportunity to participate in the project and also for the guidance and valuable advice during the concept formation of the entire application.

In addition, I would like to add my thanks to my family and friends for their care and mental support throughout my study on the CTU.

Michaela Vlčková

Title:
Content sharing in the EduARd system

Author: Michaela Vlčková

Field of study: Open Informatics

Subfield: Human Computer Interaction

Thesis type: Master's thesis

Supervisor: Ing. David Sedláček, Ph.D.

Abstract: The thesis is expanding the EduARd project features by adding the possibility to share the created content, therefore it widens the possibilities for its users to experiment with a lot more resources and to share their creations.
In the thesis, the summary of the project's current state is followed by the analysis of the similar services and dedicates a section for the analysis of the User-Centered Design methodology as the intended direction of development.
Then it proceeds to the design of the user interface for two considered user flows: The Marketplace resource addition and consumption.
It decides to implement the API as a separate service to fill the requirements without disrupting the current API implementation. The frontend section for the adding, using, listing and filtering of the Marketplace resources is integrated into the existing solution.
Based on the design, the prototype that allows the adding, listing and consuming of the resources is implemented. The thesis ends with qualitative user studies and quantitative user survey proposals.
Key words: EduARd, content sharing, ASP.Net Core, Postgres, React, virtual educational tools, user-centered design

Název práce:
Sdílení obsahu v systému EduARd

Autor: Michaela Vlčková

Abstrakt: Práca rozširuje možnosti projektu EduARd o zdielanie obsahu, čím umožnuje užívateľom experimentovať s väčším množstvom podkladov a zdielať ich vlastný vytvorený obsah.
V práci je stručný súhrn súčastného stavu projektu, nasledovaný analýzou podobných služieb a venuje sekciu analýze UCD metodiky, ktorú stanovuje ako zamýšľaný smer vývoja a návrhu.
Následuje dizajn užívateľského rozhrania pre dva zvažované užívateľské postupy: pridávanie a vyžívanie podkladov z trhoviska.
V rámci dizajnu práca určuje implementáciu API na obsluhu podkladov trhoviska ako oddelenú aplikáciu. Frontendovú časť implementácie na pridávanie, prezeranie a filtrovanie podkladov navrhuje integrovať do už existujúceho riešenia.
Na základe dizajnu je naimplementovaný prototyp umožňujúci nahrávať na trhovisko podklady, prezerať si tie už pridané a využiť akékoľvek z nich. Práca končí návrhom kvantitatívneho dotazníka a kvalitatívnej užívateľskej štúdie.
Klíčová slova: EduARd, zdieľanie obsahu, ASP.Net Core, Postgres, React, virtuálne výukové pomôcky, dizajn zameraný na užívateľa

Contents

List of Acronyms

API - Application Programming Interface

HCI - Human-Computer Interaction

UX - User Experience

UCD - User-Centered Design

CRM - Customer Relationship Management

DTO - Data Transfer Object

XML - Extensible Markup Language

FR - Functional Requirement

UR - User Requirement

List of Figures

Introduction

The thesis covers an extension to an already existing system. This first short chapter answers the fundamental questions: what is EduARd, and why the topic was created.

About EduARd

EduARd[1] is a system for developing and using teaching aids in the form of virtual educational paths consisting of the GPS bound books, tasks and 3D models created by the teachers of elementary and high schools for their students and pupils to use on portable devices. It consists of two major parts: Web editor for the teachers to create and manage all the educational paths materials and mobile applications used by students to consume prepared content.

Motivation

The creation of the resources can be challenging and time-consuming for the teachers. In most cases, the same or slightly modified content can be used by multiple institutions in various use cases. One of the ways to make the whole process easier is to allow the possibility to start from already existing content. As not all of the items are suitable for this approach, it depends on the author's judgement to decide whether the result should be published for others to take. It is the definition of the Marketplace this thesis is trying to design so the content received by students can have the highest possible quality and allow all teachers to be able to do more advanced materials to best explain and offer the knowledge on selected topics.

Chapter 1

Analysis

Aspects that are considered before the design begins are the current state of the EduARd system itself, other marketplace services used in the educational field, and the tools to approach the designing process itself.

1.1 The current state of the implementation

When writing this thesis, the system is ready and used by at least one institution. This fact provided an advantage of having existing production data used in both the design process and a test data set of the implementation.

1.1.1 Backend (API and database)

The backend part was developed as a Bachelor Thesis by Jan Skála[2]. It contains user and institution management and authentication, storage management of all the assets and source files of resources, and API for the web editor and mobile applications. Complete data structure is described in a form of relational schema in the figure 1.1.

The API uses the ASP.Net Core framework[3], built on .Net Core and C# as a programming language. All data are stored in PostgreSQL[4] database. The same technologies are used for the API of the implementation prototype to allow easier integration of the results of this thesis.

For the easy preview of the API, the Swagger API Platform[5] tool is used. It allows to visualise and test all API endpoints with its options directly in the web browser.

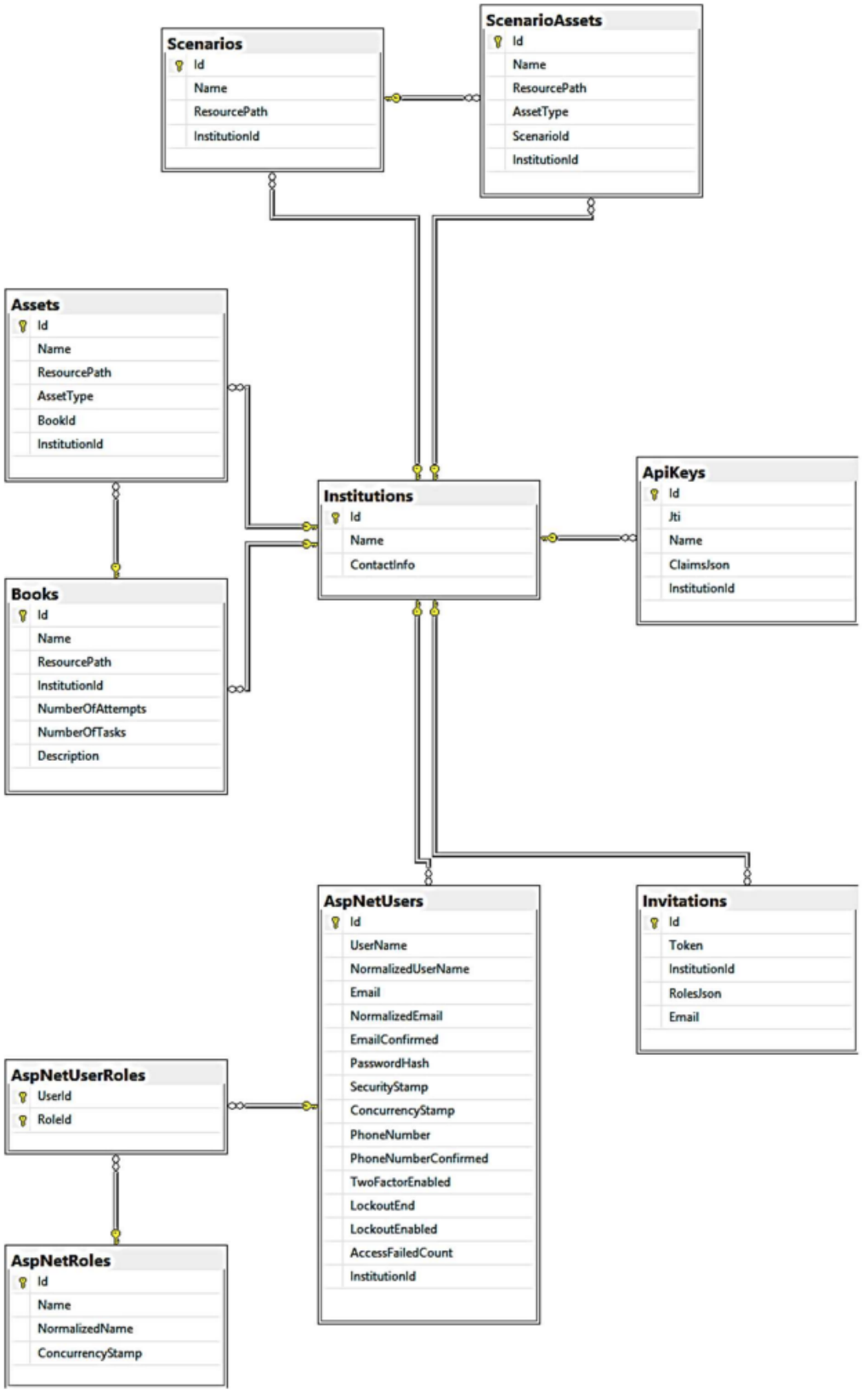

Figure 1.1: Relational schema of primary backend database.[2]

1.1.2 Web Editor for resources management (for teachers)

System for teachers was developed as a Bachelor Thesis by Anastasia Surikova[6]. It is available on `https://editor.eduard.fel.cvut.cz/` and allows the management of the educational materials, in this case only the 2D sources named in the system as Books, whose edit layout preview is shown in the figure 1.2.

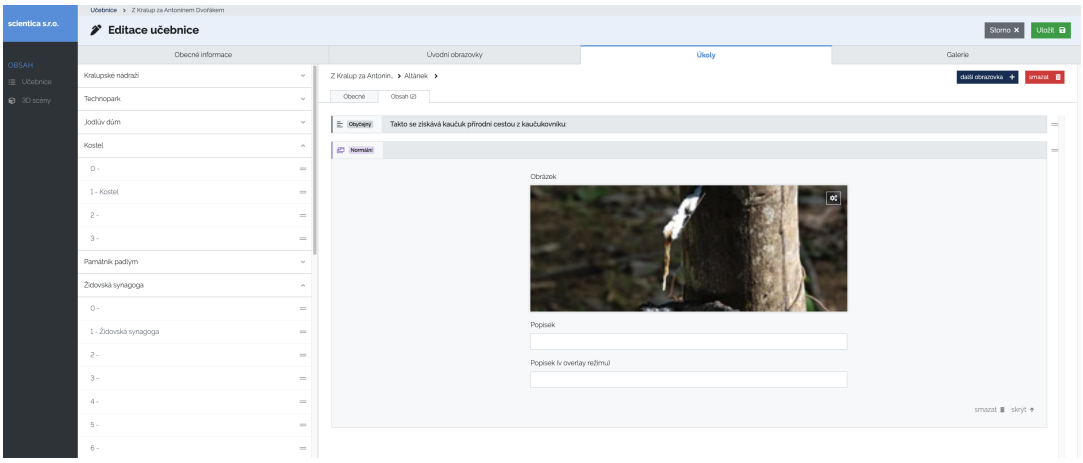

Figure 1.2: Web editor preview - Book edit.

It also contains the possibility to manage 3D scenarios and resources, developed as Bachelor Thesis by Michal Mráz[7]. This part is implemented separately and is embedded in the editor via iframe, as seen in the figure 1.3. It is hosted at `https://ar.eduard.fel.cvut.cz/index.html` and does not use the same front-tend framework as the rest of the web editor. For this reason, the implementation prototype of this thesis focuses on the Books, but since the source files of both types of content are represented in XML files, it should work with both types of content. Though, this statement is not tested.

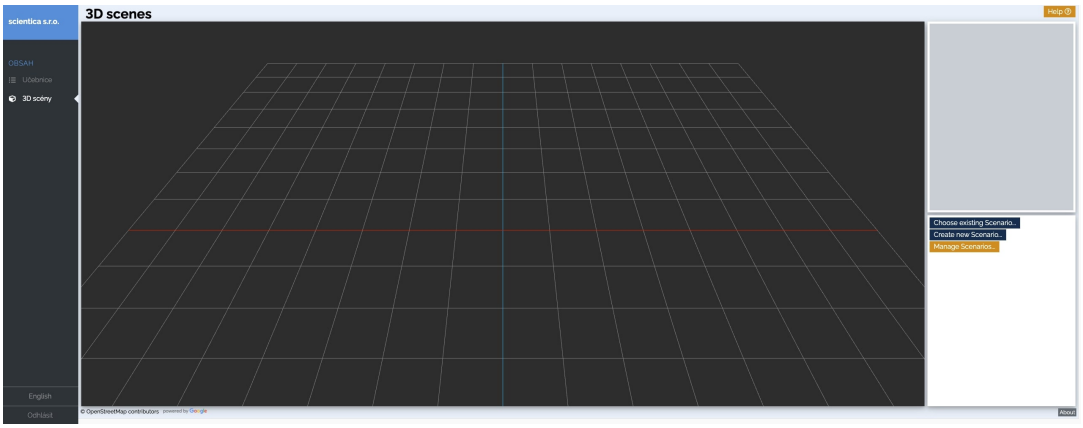

Figure 1.3: Scenarios embedded section preview.

1.1.3 Mobile applications

Multiple mobile applications are developed for content consumption. They vary in the used frameworks and target platforms, for example Java[8] and React Native[9], etc. The one written in React Native is currently available on AppStore[10] and GooglePlay[11] under the name "Stezkami vědy"[12]. It was developed as a Bachelor Thesis by Šimon Maňour[13]. The figure 1.4 shows a few screen examples available in the app description on mentioned stores.

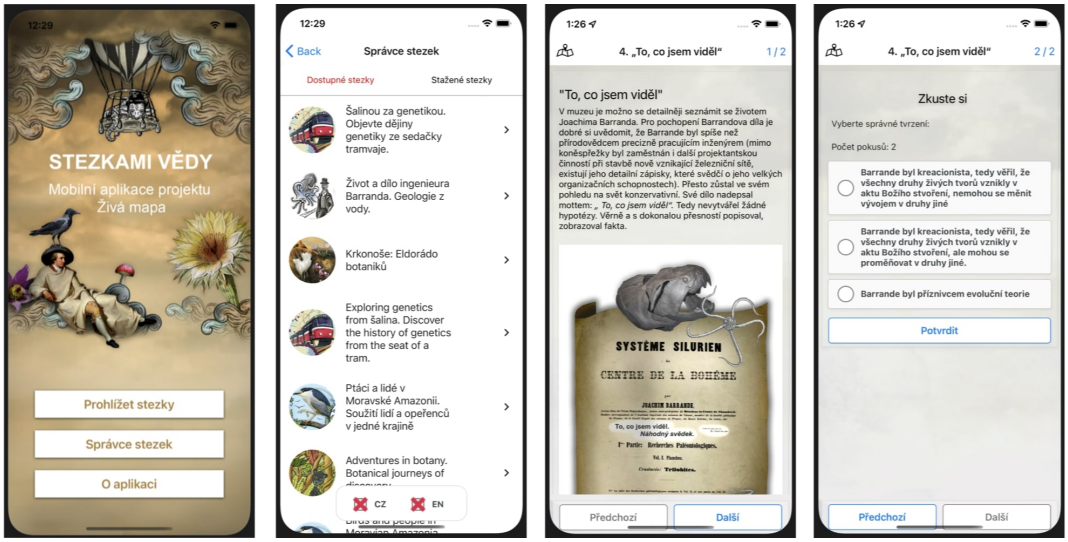

Figure 1.4: "Stezkami vedy" mobile app preview.

All mobile applications use the API provided by the backend mentioned in the section 1.1.1.

1.2 Application of user-centered design principles

The design process takes advantage of the User-Centered Design (UCD), specifically, its base principles summarised in the book User-Centered Design: A Developer's Guide to Building User-Friendly Applications written by Travis Lowdermilk[14]. This part states which specific processes were considered and how they were addressed in this particular case.

1.2.1 UCD definition and its distinction between other commonly used user-focused terms

As the terminology around the UCD can be confusing, the thesis's primary definition is from the mentioned book. User-Centered Design (UCD) emerged from HCI and is a software design methodology for developers and designers. Essentially, it helps them make applications that meet the needs of their users[14]. Key focus is that the

definition includes developers, so even when the visual concepts cannot be neglected, they are not in the front and centre of the design process. The figure 1.5 from the book visualizes defined scope in the whole process of creating application of certain usability requirements.

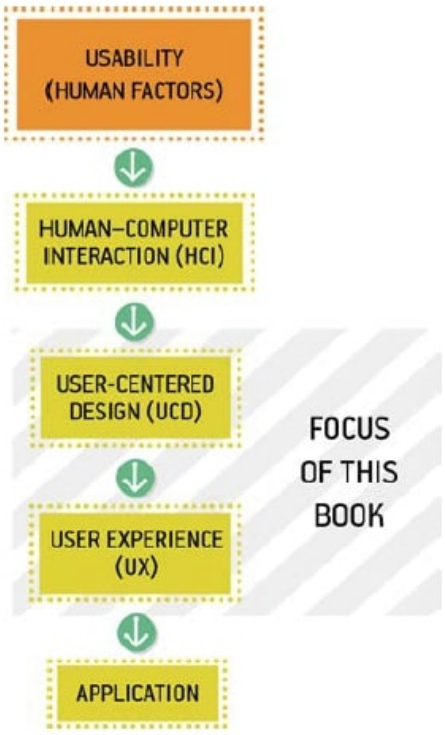

Figure 1.5: Simplified graph of UCDs position between other user-centered terminology.[14]

1.2.2 UCD specific points and procedures applied

The most commonly known rule of the UCD is the inclusion of potential users in the whole design process from the beginning. This rule is slightly bent, as the primary source of feedback prior to the first implemented prototype is the semi-week consultations with the thesis supervisor, as he is the closest person to the target audience available. Nonetheless, other users are sporadically interviewed to ensure optimal results.

Another process taken from the methodology is to include the impasses that were abandoned in the design but provide some crucial information to the final result.

As the product of the thesis is not a standalone project but the extension of an already existing one, the rule of taking into account already implemented, tested and successful functionalities is also widely interpreted. Same stands for the principle of following a specified manifesto and major goal, which the whole system tries to achieve.

On top of the mentioned rules, major design principles widely used, well defined and based on extensive research are utilised throughout the design chapter.

Major design principles used across the whole design process:

- **Proximity Principle** - grouping components, if they are related or have similar functionality.
- **Visibility** - using the colour, opacity, and prominence attributes to guide users' focus in desired flow.
- **Visual feedback** - after each user interaction, immediate visual feedback is shown to let the user know that the application registered said action.
- **Mental models and metaphors** - use commonly known processes and icons instead of re-inventing ones to make the flow as natural and easy as possible.
- **Progressive Disclosure** - making available only viable options in any current state.
- **Consistency, Affordance, Constrains, and Confirmation** - are all principles to ensure the user flows seamlessly and avoids unwanted changes and other errors.

Multiple sketches and wireframes preceded the extension's code implementation, focusing on the goals of the UCD rather than its theoretical definitions.

1.3 Other existing content sharing services

The principle of sharing existing content, especially when the content is usually highly structured and time-consuming to prepare from scratch, is extremely common among the vast majority of fields. To gain more inspiration, shallow research of existing and functioning marketplace services occurs.

Hubspot

Hubspot[15] is a CRM containing the features of creating websites, marketing and transaction emails, forms, marketing campaigns and many others. The analysis is done on their Asset Marketplace, where users share and consume various themes, website and email templates, and modules (building blocks of aforementioned).

From the structure and complexity point of view, website themes are more similar to EduARd Marketplace items than singular templates. Hubspot introduces two layouts for asset detail pages, so only the one for themes is included. The layout is focusing on the most visual and the most important part as the same time, with all the info centered about the use of the theme on the side, where it is not distracting, but it is still above the page fold as seen in the figure 1.6.

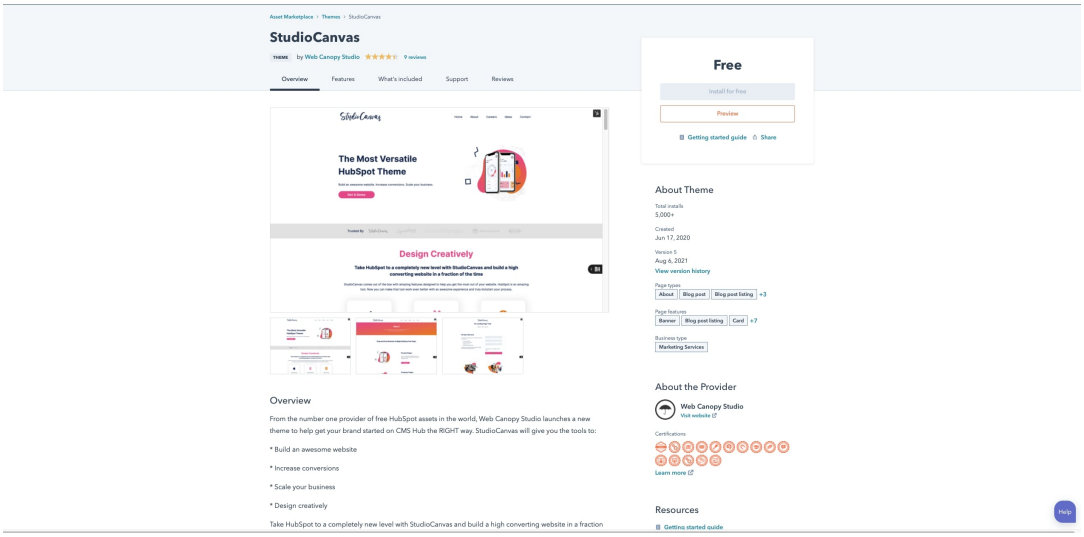

Figure 1.6: Screenshot of Hubspot Asset Marketplace Asset Detail.

Features considered for further exploration:

- Showing structured detail information, as well as a more extended textual summary of the asset;
- Extended information separated into multiple sections, with anchor links for each at the top;
- A clean layout with the visual preview gallery in the centre of focus.

1.3.1 Unity Asset Store

When considering fields where the assets have complex structures, even when the space is not strictly speaking a marketplace, Unity Asset Store[16] provides much insight into successfully creating a UI that incorporates previews of many different types of content by including relatively simple components.

As seen in the figure 1.7, the main layout and content are very similar to the one provided by Hubspot, with videos as an item of the preview gallery. This small change opens significant expansion on the possibilities of the asset presentation.

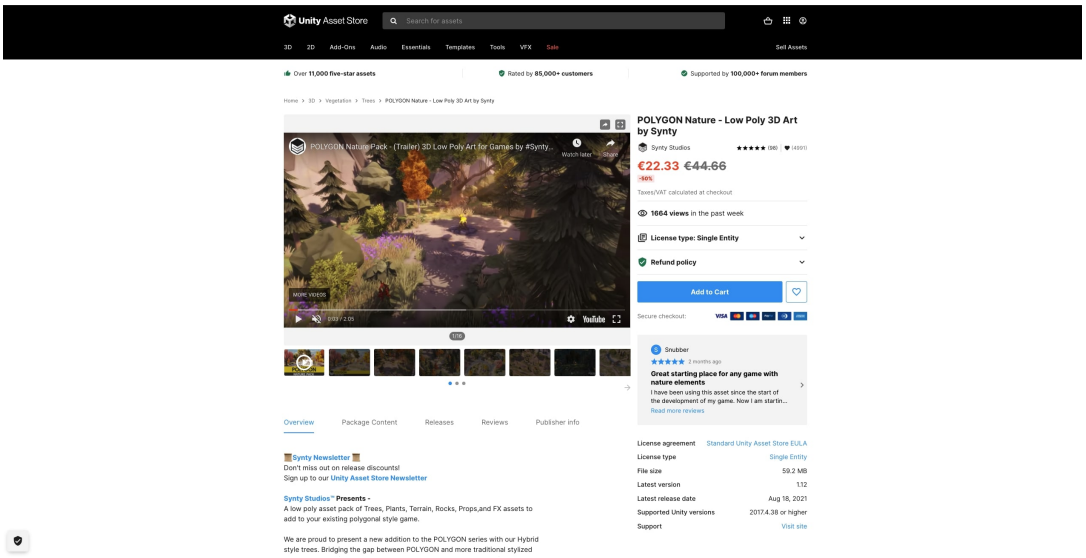

Figure 1.7: Screenshot of Unity Asset Store - Asset Detail.

1.3.2 Sketchfab

Sketchfab has aimed to empower a new era of creativity by making it easy for anyone to publish and find 3D content online[17]. By creating a vast community of users and providing exceptional UI components to present 3D content, they took the preview to a whole other level.

As the content is about 3D, a custom interactive component allowing users to see promoted models from all angles is a central focus of the detail. The figure 1.8 shows controls of this complex component. This level of complexity for preview might be a little extreme for the studied project, but in addition to this component, it shows some other differences to other researched spaces. Detail preview is sometimes displayed in the modal window but shown as standalone page when reloaded or shared by URL.

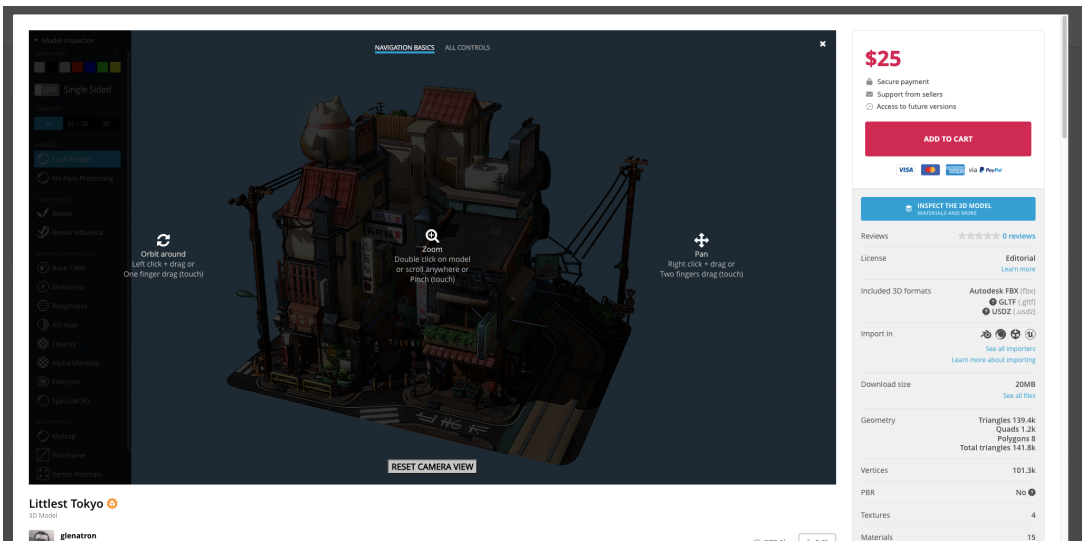

Figure 1.8: Screenshot of Sketchfab Detail.

Minimalistic filters are placed them on top of the viewport, allowing the users to have as much space as possible to focus on visual aspects of previewed models, as seen in the top section of the figure 1.9.

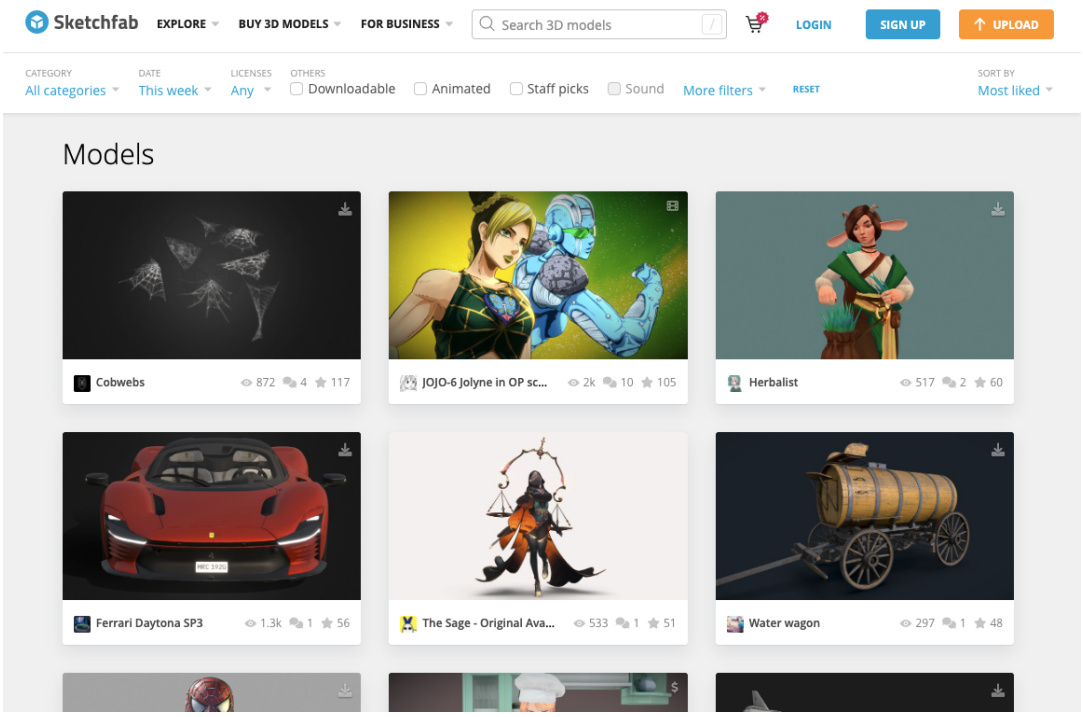

Figure 1.9: Screenshot of Sketchfab Homepage.

1.3.3 Merlot

The MERLOT system provides access to curated online learning and support materials and content creation tools led by an international community of educators, learners and researchers[18].

It is the only strictly educational system researched, and the first observation is that visually, it does not have such an appealing and clean design as the other services mentioned in this section. However, apart from this fact, it follows similar patterns seen in the commercial products. Layout of the list of searched results is pretty much the same, and the detail view is only missing the centred visual preview component, which is to be expected as the content published here is mainly in the form of PDF files.

An interesting component is a social rating system, where the ratings are separated into peer review (the teacher to teacher reviews) and user rating (the feedback from the content consumers), as visible in the top right corner in the figure 1.10.

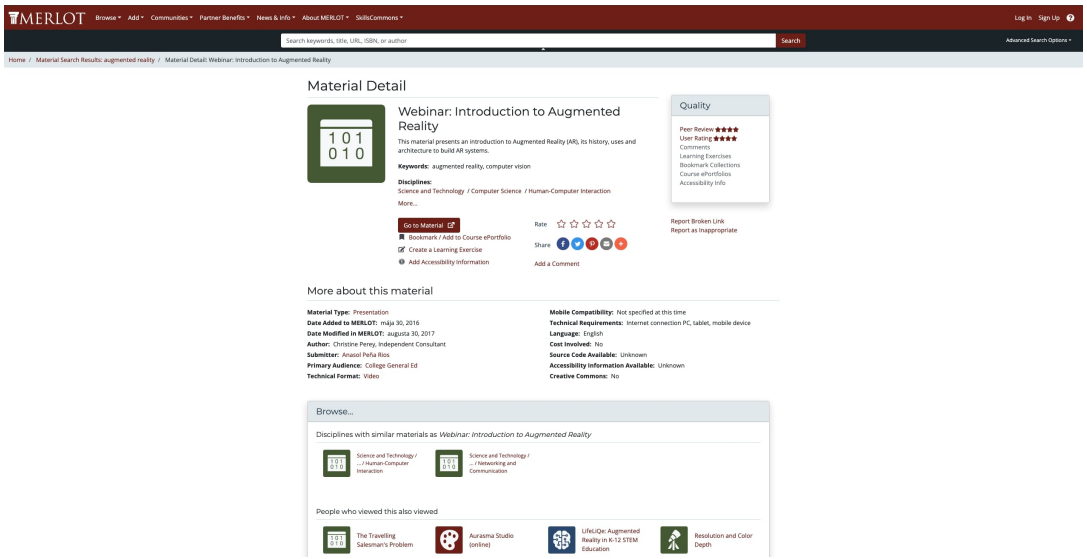

Figure 1.10: Screenshot of Merlot Detail.

1.4 Findings summary

The design chapter will mainly focus on fulfilling these points:

- Respect the methodology of UCD and follow the processes specified in the section 1.2.2.
- Follow already tested and proved design principles.
- Take advantage of the already existing components to follow the consistency principle.
- Incorporate methods and functionalities of other similar services that will benefit the final UX of the extension.

Chapter 2

Design

The main goal is to help teachers create helpful and extensive teaching aids by allowing them to start the preparation of the resources by using already existing ones. It is be achieved by integrating a new section to the book management to allow users to publish their results to the Marketplace and by adding the Marketplace environment itself with the listing of all the shared resources with filtering, sorting and previews, and by adding the possibility to use selected resource within the current institution.

First, the requirements to cover all the user flows are specified, then they are converted to the use cases and designed in wireframes for each necessary step. The last section of the chapter specifies the technical definition of the system architecture and all the aspects from the implementation point of view.

2.1 Requirements

The user requirements set the foundation for the remaining steps in the user-centered design process[14]. A user requirement is what the user needs; a functional requirement is what the application needs. Essentially, functional requirements can be seen as the technical specifications of the project[14]. To follow best practices, first, the user requirements are stated from a non-technical point of view and are followed by functional requirements that implement them.

2.1.1 User requirements

The list of requirements is specified after numerous consultations and some informal interviews with active clients of some of the existing services mentioned in the analysis chapter. All the requirements are expanded further in the design chapter.

Specification of user requirements:

1. See what is already available for use.
2. Sort the listed content to see the best suitable items first.
3. See all available details of the picked Marketplace resource.
4. When decided, quickly start customising the chosen resource from Marketplace.
5. Publish created content for others to use.
6. Hide the content that was published before.
7. Provide more information about the content that is being added to the Marketplace.
8. Preview how the content will be seen on the Marketplace.
9. Ensure that the resources published on Marketplace by one user stay editable by just the user that published them.

2.1.2 Functional requirements

Functional requirements are technical definitions of all the functionalities necessary to cover all user requirements.

1. In Marketplace section:
   (a) list all the resources published on the Marketplace.
   (b) sort resources by the rules specified in the UI design.
   (c) filter resources by 1-N constraints specified in the UI design.
   (d) reset filters, if some are selected.
2. In the extension of existing item edit, allow the user to:
   (a) publish it on the Marketplace.
   (b) fill in detailed additional information about the item specified by UI design.
   (c) unpublish the item from Marketplace.
   (d) edit detailed additional information.
   (e) claim the item. The current user will be set as an author, and other users in the institution, except the admins, will not edit the published Marketplace resource.

3. Marketplace resource preview contains all the detailed additional information the user fills in the item edit extension. Make preview available:
   (a) as a separate page visible after the interaction with Marketplace resources list item.
   (b) as the embedded component available while editing detailed additional information in the item edit extension.
4. Use an item from the Marketplace in the current institution.
   (a) Copy all the source files from the Marketplace to the current user's institution.
   (b) Redirect the user to the newly created item edit section to allow further customisation.

The table 2.1 connects all functional requirements to the individual user requirements.

User Requirement Number	Implemented by the functional requirements
1	1.a.,1.c.
2	1.b.
3	3.a.
4	4.a.,4.b.
5	2.a.
6	2.c.
7	2.b.,2.d.
8	3.b.
9	2.e.

Table 2.1: Matching a functional requirements with the user requirements they meet.

2.2 Design of the user interface

The UI design will expand user requirements into use cases and explain the steps to create the individual screens in the form of wireframes created in the Balsamiq Cloud[19]. As the visual theme of the editor is already set, the development will skip the transition of the wireframes into the visual design.

2.2.1 Entities

The section specifies entities in a sort of loose interpretation. It lays the base for the database design, but the final schema may slightly differ to accommodate all needed relationships and dependencies.

1. **Resource** - Marketplace item of any type (Book as 2D resource or Scenario as a 3D resource).

2. **Filters** - Properties with a specified array of possible values.

   (a) **Property** - Defined in the database, therefore somewhat dynamic, but its management will not be included in the web editor.

   (b) **Value** - Paired with property, selectable and allowing the filtration in the item list.

3. **Screenshot** - Bound to resource, previews how the item looks like in the mobile app for students.

4. **Asset** - Bound to resource, any physical file included in the original resource that is copied to marketplace storage.

2.2.2 List, sort and filter Marketplace resources

Covered use case (merging user requirements 1 and 2): The teacher wants to prepare new content for students. They already have a basic idea of what the new path should cover. Before they start, they visit the Marketplace section of the editor. They will sort all the resources by a suitable attribute and filter them based on the intended content and target audience.

The editor already contains almost all of the necessary components (except sort). So the first idea was to repurpose as many of them as possible and design only the missing ones. During the consultations, brainstorming showed that this approach would hurt the usability more than help. Mainly from the point of view that it might be confusing for users to orient if they are listing the books already in their institution or browsing the Marketplace.

Another important reason to redesign the Marketplace list item component specifically is that there is a need to show much more information than is already present in the Book list item. After a few sketches, this components is designed from scratch but as to resemble the existing one, so the consistency principle is not broken.

List item component

The layout is flipped with the main image on top to avoid confusion with book list items. This flip also provides more space to add base information about the resource as seen in the figure 2.1.

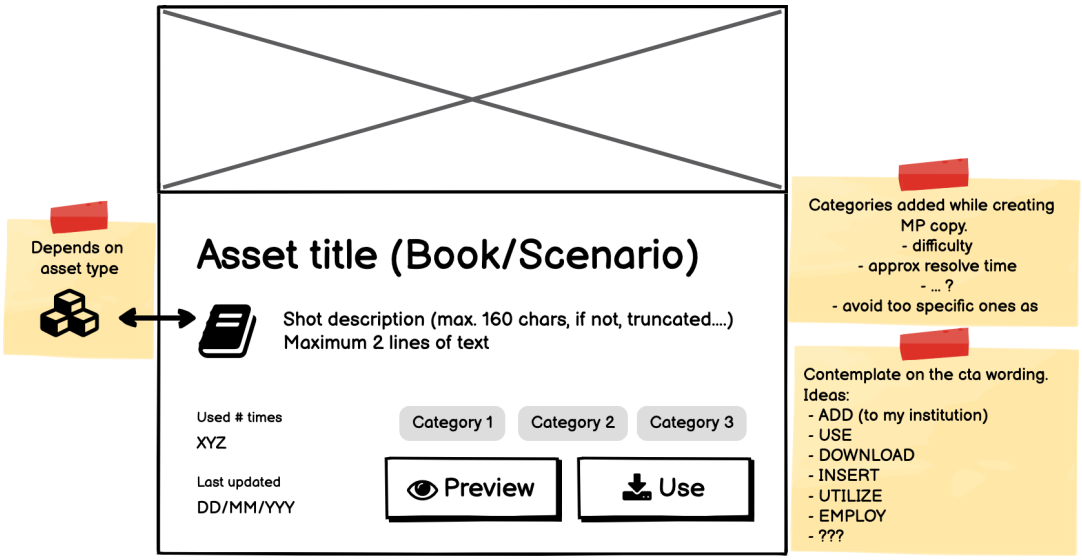

Figure 2.1: Marketplace list item wireframe.

The list item card component summarises most of the information, by which the user sorts and filters it and provides an overview when seen in the list. Users then can easily compare similar resources without the immediate need to see detailed previews.

One specific use case is discussed when contemplating the similarities during the publishing process. The user just used some Marketplace item and changed nothing, or very little. As creating the compare functionality and the functionality of resolving, if the new item is sufficiently different from the original Marketplace resource would be way over the scope of the assignment, an alternative approach is designed. In this case, during the publish of an item to Marketplace based on another Marketplace resource, the user is prompted by a question in the modal window asking him if the item should be published as a new resource or as a variant of the original Marketplace resource. If he decides to publish the resource as a variant, all variants of the same resource fulfilling the selected filters will be grouped into one list item card as depicted in the figure 2.2.

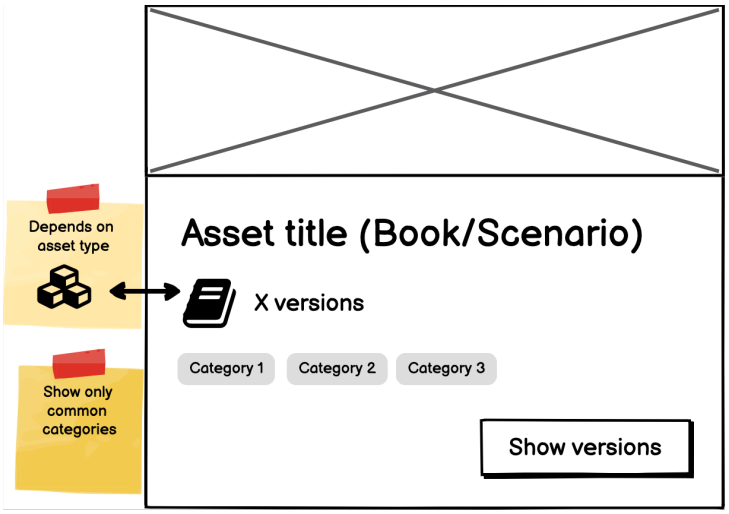

Figure 2.2: Marketplace list item with multiple versions.

This use case is considered an additional feature, and the rest of the design does not cover all the aspects that this feature will require. It is included in the ideas for future development.

Filters component

The existing filtering component used in the book list allows only filtering by language and is a single choice. From the current dataset, it is clear that one resource can cover multiple values of any property. So the new multi-choice filtering component is introduced as seen in the figure 2.3. This component will be reused in the use case of adding an item to the Marketplace.

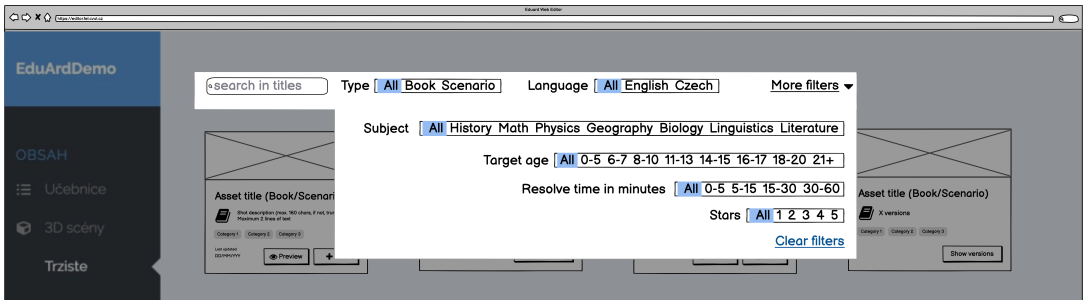

Figure 2.3: Marketplace filters component wireframe.

Properties and their possible values should be expanded over time, but to cover as much of the current content as possible, here is a list of those added by this design:

- Target age:
  - 0-5 / 6-7 / 8-10 / 11-13 / 14-15 / 16-17 / 18-20 / 21+ / No limit
- Subject:
  - History / Math / Physics / Chemistry / Geography / Biology / Languages / Literature / Other (Not specified)
- Approximate resolve time:
  - 0-5 minutes / 5-15 minutes / 15-30 minutes / 30-60 minutes / more than 1 hour / Other (Not specified)
- Language (already existing information):
  - English / Czech / All
- Content type (filled automatically, based on the location the user is publishing the content from):
  - Book /Scenario / All

- Stars:
  - 1 / 2 / 3 / 4 / 5 / All
  - This attribute will be filled after some users that already used the resource rate their experience with it. They will be prompted to rate such resources after a specific time via email. If they choose not to fill the review, they will not be prompted repeatedly.
  - The process of rating will take place directly in the email template. They will have the option to click on one of the values, which will generate a request with the resource id, value and user id to prevent multiple submissions by one user. This is considered as feature proposed for future implementation and is not included to the implementation of the prototype.

In addition to the categorisation filters, a text search field is added to filter items based on the textual content of the resources.

In this scope, the dynamic addition of filters is possible only by manipulating entried in the database directly. In the future development, this management could be added to the administration of users and institutions developed by Jan Skala in his thesis[2].

Sort component

Based on available data about the resource, available attributes are to sort by last updated, most popular (reflecting number of usages), best rated and best match according to selested filters. Default is by the best match, and the component is implemented by default select input as seen in the figure 2.4.

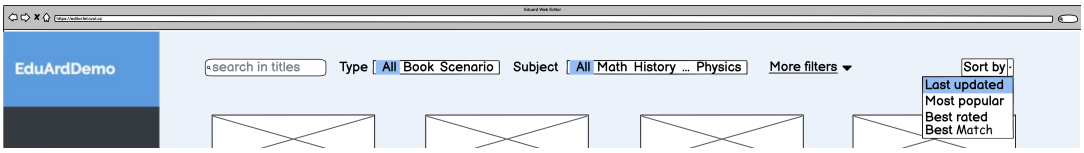

Figure 2.4: Marketplace sort component wireframe.

The Marketplace section

All other components used in the Marketplace section, such as buttons, and pagination are repurposed from the original implementation for the book list as depicted in the figure 2.5.

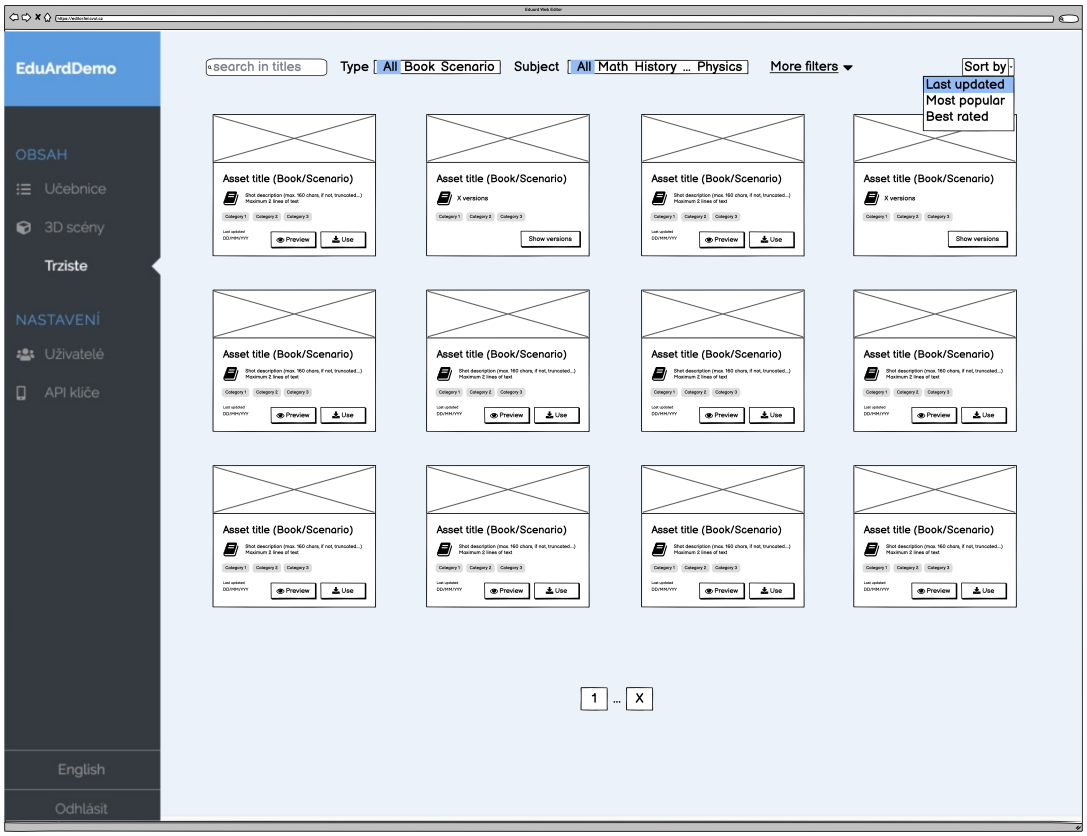

Figure 2.5: Marketplace resources list wireframe.

2.2.3 Manage the Marketplace resource

Covered use case (applying user requirements 5-7 and 9): The user just finished the educational path for the students. They put much time into this and realise that it covers quite a general topic, which could benefit other teachers outside of their organisation. They decide to publish it on the Marketplace. They claim the content to be marked as the author and allow the publishing. They will fill in a more detailed description, mark categories into which the educational path falls and upload a couple of screenshots from the application the content is intended for. After some time, they edit the educational path. When saving, the system detects that it is published on the Marketplace and informs about the changes of the Marketplace resource. If another colleague from the same institution updates the book, no prompt is displayed, and they cannot update nor synchronise the Marketplace resource. After some more time, the author decides that the content went out of date and will unpublish the resource from the Marketplace.

The early stages of this section of the design considerations include only the "Publish to Marketplace" switch and operating on the Marketplace resources with the information available by default. Given that the students consume content differently from the teachers, a decision was made to add extended information to draw a better picture of the used resource from both points of view. That means the expansion of the book edit with a new Marketplace tab and filling it with the inputs that summarise the content in general.

Claim the resource

Before the book is published to the Marketplace, the "Published" switch is set to "No", and all other fields on the Marketplace tab are disabled. After clicking "Yes", the user is prompted to confirm that he is an author and will manage a new Marketplace resource. After this action, a Marketplace item is created and bound to the current user. When anyone else tries to update this tab, all the fields are disabled, and they are informed that only an author can update this section.

To ensure that the Marketplace resource is still editable when the author is no longer part of the organisation, admins can change the author to another user or "none". When the author is set back to "none", the Marketplace item stays intact, but the warning is shown, and any new user editing the book can claim it. The Marketplace resource will also be deleted when the original book is deleted. The user is informed about this action when the confirmation dialogue for the book delete is displayed.

Components on the Marketplace tab

Existing components populate the tab:

- Publish to the Marketplace switch (copying the "Published/Not published" switch from the General Information tab), with additional functionality explained in the Claim the resource section.
- WYSIWYG editor[20] allows the user to include a more extended resource summary.
- Categorisation section - using the same component designed in the section 2.2.2.
- Screenshots - using the gallery component, simplified version of the component used on Gallery tab.
- "Preview" button - showing the Marketplace resource preview component designed in the next section.

As an additional feature to make publishing more effortless, the resource summary WYSIWYG editor[20] has default content generated from the available information about the book. The user can use the generated summary or update it to their liking.

The layout of the Marketplace tab is shown in the figure 2.6

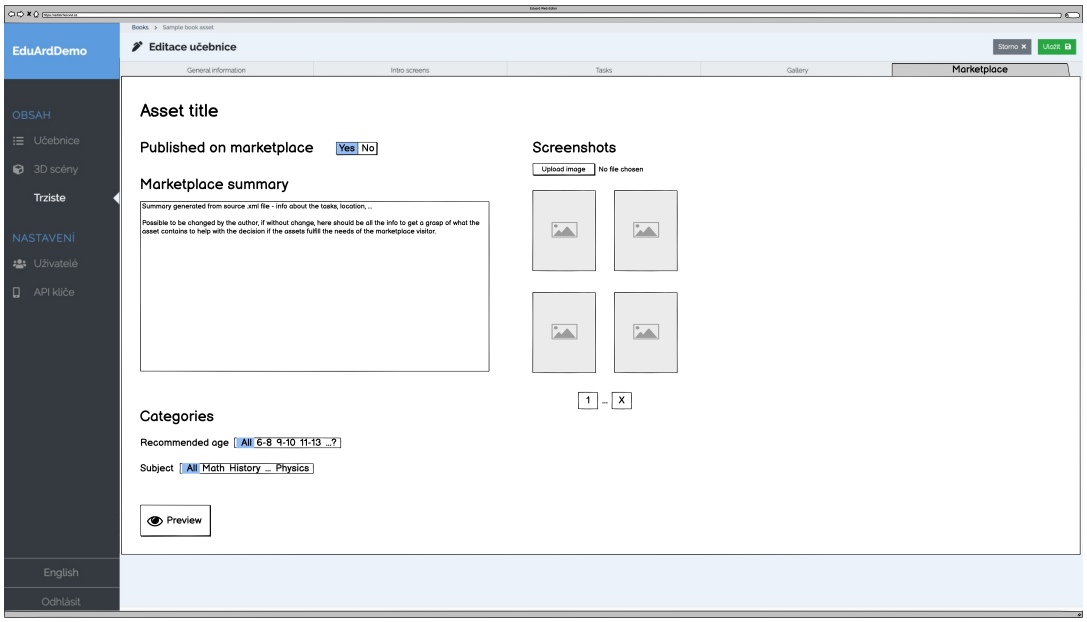

Figure 2.6: Book edit - publish to the Marketplace wireframe.

Generated default resource summary structure template in pseudo code:

```
<p>The resource introduces users to the addressed topic on
${numIntroSlides} intro screens.</p>

<p>Then they are presented with ${tasks.length} tasks.</p>

<p>The average number of screens per task is
${Math.avg(slidesPerTask)} where the minimum of screens is
${Math.min(slidesPerTask)} and the maximum is
${Math.max(slidesPerTask)}.</p>

<p>Question types used in the tasks are:</p>
<ul>
  <li>Numeric answer <strong>${QUESTION.Numeric.length} x</strong></li>
  <li>Toggle <strong>${QUESTION.Toggle.length} x</strong></li>
  <li>...</li>
</ul>

<p>Throughout the tasks, students are presented with following
media types:</p>
<ul>
  <li>images: ${Assets.length}</li>
  <li>videos: ${Videos.length}</li>
  <li>audio examples: ${Audios.length}</li>
</ul>

<p>Number of screens with 3D content is ${slidesIs3D.length}.</p>

<p>Tasks are bound to the following GPS coordinates:</p>
<ul>
  <li>
    <a href="https://www.google.com/maps/search/?api=1
        &amp;query=${task.lat},${task.long}">${task.lat}, ${task.long}
    </a>
  </li>
  <li>...</li>
</ul>

<p>Number of tasks with the assigned info about CloudAnchor
is ${cloudAnchors.length}.</p>
```

If any of the values is zero or non-existent, the whole part is excluded.

2.2.4 The Marketplace resource preview

Covered use case (including user requirements 3): The user spent some time browsing available resources and decided on the one they recon would be most suitable. They click "Preview" on the resource list item card and are presented with all the information the resource contains. They see all the general information, screenshots from the mobile application, resource summary and values of all available categories.

All the information available for any resource is:

- Resource summary
- General information from the tab with the same title on book edit.
- Screenshots from the mobile application, if provided by the author
- Resource categorisation

The component tries to accommodate a lot of information into one space, so the expanding sections are introduced into the design. The general information section is expanded by default, and the user can click through the other sections. The only exception is the resource summary, recognised as the most informative and, because of that, is positioned statically under the resource title. 0-N sections can be expanded at one time, so the user can see all the information available if wanted, as depicted in the figure 2.7.

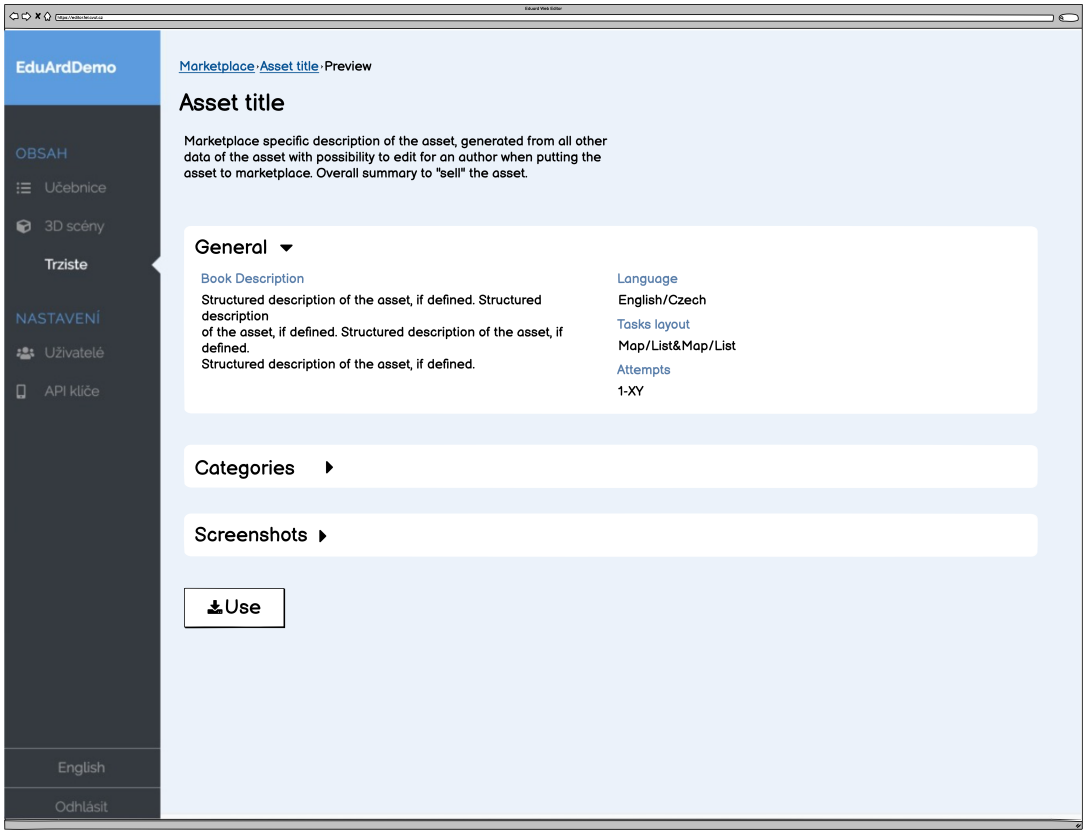

Figure 2.7: The Marketplace resource preview.

The resource author is presented with the same component when the "Preview" button on the Marketplace tab is clicked (only with the "Use" button missing) to check the published resource's final presentation.

It was considered to follow a more traditional layout of the Marketplace item presented in the Analysis chapter. However, the screenshots section is optional, and from the available data, it is easy to assume that the images are not always an essential attribute of the content. Therefore the final layout follows more linear lines putting the textual summary first.

2.2.5 Use a Marketplace resource in the current institution

Covered use case (in line with the user requirement 4): The teacher decided to use one of the resources from the Marketplace. They will click the "Use" button and proceed to customise the newly acquired educational path.

The user starts this flow by clicking on the "Use" button available on the preview or the list item card. They are presented with the prompt asking them to fill the new resource title as a simple way to allow the user to find the new item in their institution afterwards. After the title change and clicking confirm button, the user is redirected to the edit of the newly created book, where they routinely continue the content preparation.

2.3 Architecture design

This section states the chosen development architecture to specify the implementation scope and show the technical aspects of the prototype.

2.3.1 Standalone Marketplace Backend

Primary Backend implementation is an extensive structure with many features and functionalities. One of the most important is the encasement of data and assets within the institution. It follows strict security standards to ensure that access to all data is restricted only to the user's institution scope.

By definition, the Marketplace is going against the encasement by making the resources public for all the Web editor users. Instead of creating an even more complicated structure, the Marketplace extension adds a separate backend application.

This approach, similar to microservice architecture[21], allows adding all necessary functionalities with a relatively simple backend application. It omits the institution's authorisation restrictions to the institution's scope and allows to see any published resources.

It is expected to run parallel to the primary backend and is written in the same Asp.NET Core[3] framework. It uses a LinqToDB database access library[22] which provides a simple to use layer between the database and the application.

The source structure is simplified to:

- **Data** - mapping the database objects to the C# collections.
  - **Entities** - represent database entities and their associations.
  - **DTOs** - used to limit data transfer volumes.
- **Repositories** - collections of methods grouped by entities that implement the features and functionalities.
- **Controllers** - collections of methods grouped by entities that provide API breakpoints

The standalone backend is also working with separate local storage. When a new resource is added, the XML source file and all attached assets are copied to a separate location. This approach will ensure that the marketplace resource will not change even when the original book is updated until the user does not synchronise the resource on the Marketplace tab. All files are renamed with the unique id prefixes and then renamed again when used to avoid the file names duplicities.

Marketplace Database

A part of the standalone backend is a separate database containing just the data from the Marketplace resources and a separate location for the source files and other assets of the resources published to the Marketplace. Relational schema of the database is shown in the figure 2.8.

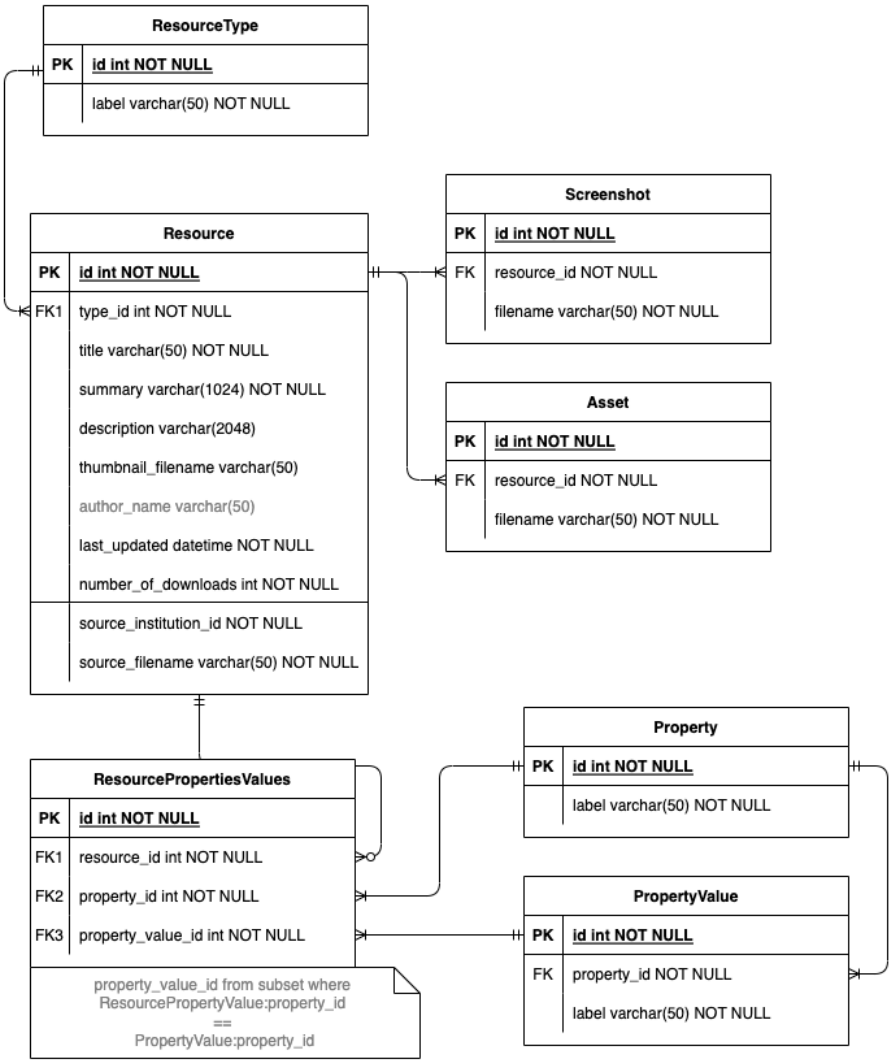

Figure 2.8: Marketplace database relational schema.

2.3.2 Integrated sections of Web Editor

The integration of the frontend matters of the Marketplace is much more intertwined with the existing editor. Therefore the frontend architecture of the Marketplace is integrated directly into the existing source. A new API connection is created in the existing editor and complements the existing structure and functionality.

Chapter 3

Implementation

A basic prototype including the main user flows and the designed structure of the extension is implemented. It fulfils all compulsory functional requirements, but, to follow the UCD principles, should be considered a powerful tool for testing and further implementation rather than a final production-ready product.

First, this prototype should be released as an experimental feature for a group of users interested. They would then be asked to participate in the further rounds of the testing, resulting in expansion of the design and implementation. The user testing design (quantitative and qualitative) for gathering feedback from such a group is a part of the testing chapter 4.

Table 3.1 summarises all the defined functional requirements and their status in the prototype.

3.1 API

The backend development follows the process stated in the design section 2.3.1. All containing API endpoints support specific functionality of the frontend. To visualise and test created endpoint the Swagger[5] tool is used, same as on the main backend API. The list of all endpoints is seen in the figure 3.1.

FR number	Is Compulsory?	Is Implemented?	Note
1a	Yes	Yes	
1b	No	Yes (Partially)	Missing the option to sort by review.
1c	Yes	Yes	
1d	No	Yes	
2a	Yes	Yes	
2b	Yes	Yes	
2c	Yes	Yes	
2d	Yes	Yes	
2e	No	No	
3a	Yes	Yes	
3b	No	No	Not marked compulsory as user can still access the preview on the Marketplace directly.
4a	Yes	Yes	
4b	Yes	Yes	

Table 3.1: Functional requirements fulfilment summary

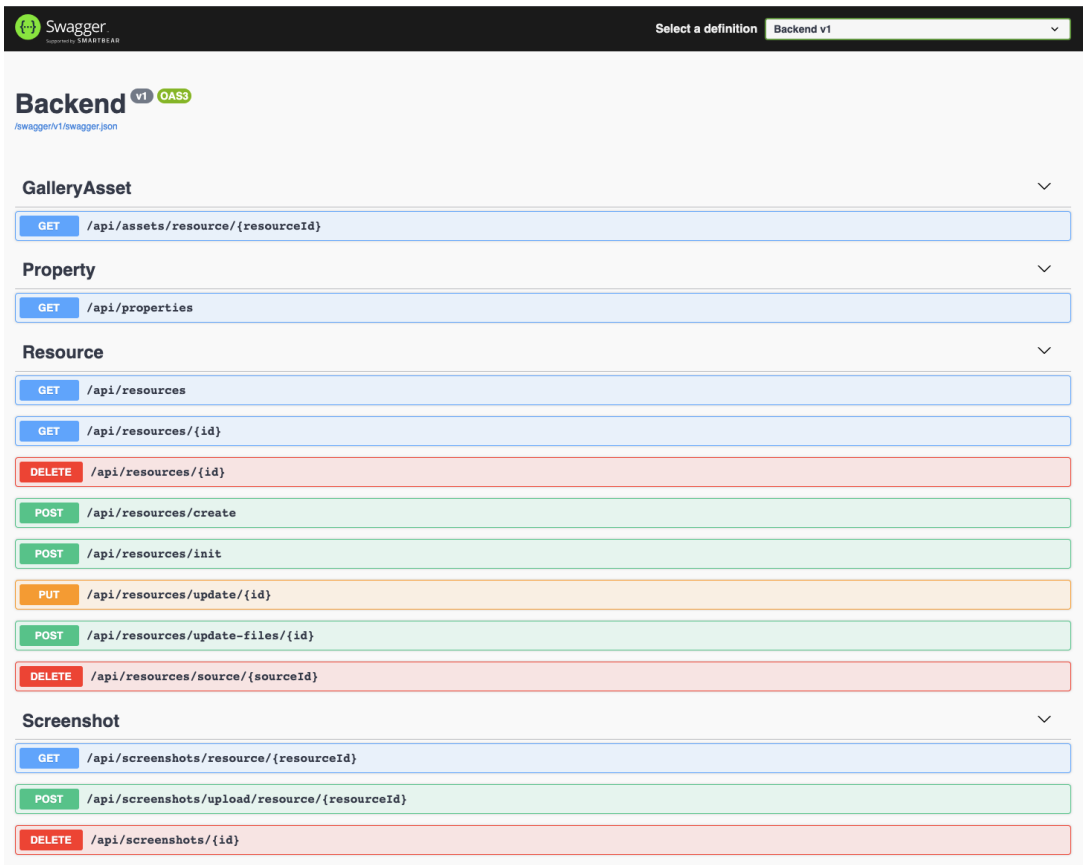

Figure 3.1: All API endpoints serving the new sections of Web Editor.

API implementation is encapsulated in Docker compose[23], together with the database. The SQL script[24] containing the database seed can be found in the root directory

of the application source code. This combination ensures a smooth process of initializing, building and running the API.

The project is available on the faculty gitlab[25] repository `https://gitlab.fel.cvut.cz/eduard/eduard-marketplace.git`.

For the project to work with the main backend, a few edits are made in the source code of the main backend as well. Changes are pushed to the branch `mp-fixes` of the main gitlab repository located at `https://gitlab.fel.cvut.cz/eduard/backend.git/-/tree/mp-fixes`.

3.1.1 Resource controller

To support easy manipulation on the frontend side, the creation of the resource is implemented as two API endpoints. `InitResource` and `UpdateResource`. In the initialization step, only a minimal DTO is transferred and added to the database to receive an id. Created id is then used in the second step when the update function is called to create all necessary data bindings, manage any files and fill all additional information provided. The later step is also used separately as a common update function.

`DeleteResource` function is called each time the resource is unpublished, as well as when its parent Book is deleted. It deletes all database entries as well as all files added during the update of the resource.

3.1.2 File management

Management of any physical files (screenshots, gallery assets and XML source files) is taken care of by `PhysicalFileProvider`[26] which gets the required file storage location from the configuration, creates the structure if needed and takes care of uploading and deleting all the files. Any action done by this class is served by `LocalStorageRepository` that is used by the `ScreenshotController`, `GalleryAssetController` and `ResourceController` (when manipulating XML source files). Files are all saved in one folder, and their name gets prefixed with resource id, to avoid errors due to cross resource filename duplicities. Filename duplicities within the resource are already avoided in the original implementation. Any update to the file is implemented as the deletion of the original file following the upload of the updated one.

3.2 Database

The final database structure is shown in the figure 3.2, which contains the relational schema of the final implementation generated by the Rider IDE[27]. Tables were

slightly adjusted against the design to allow clean and effective implementation of the requirements.

Provided seed `mp-db.seed.sql` contains the initialization of properties and their values in addition to the shown database structure.

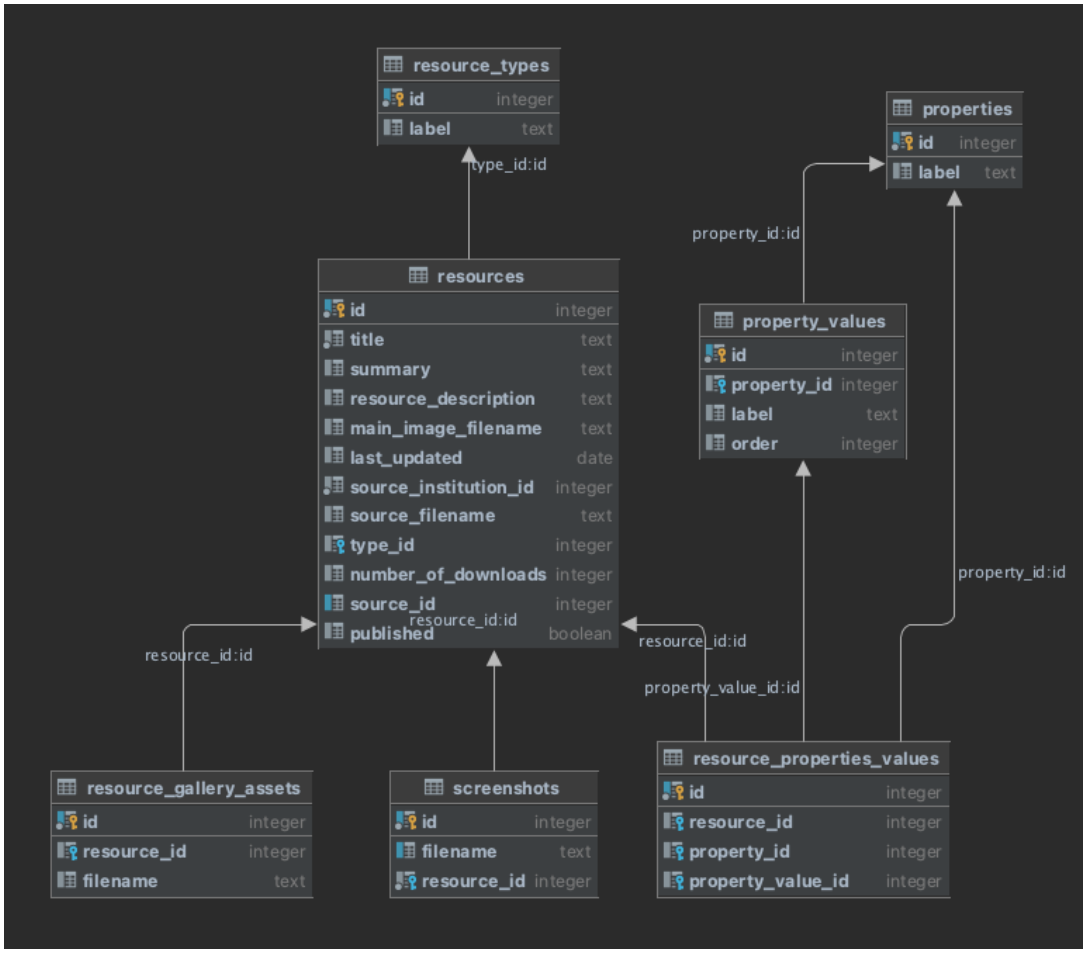

Figure 3.2: Final database schema diagram generated by Rider IDE[27].

During the integration of the frontend implementation, several additions and changes are made in all parts of the source code, but the main added functionalities are served by the components `MarketplaceItemAdd`, `MarketplaceResourcesList`, `PreviewPage` and `UseResourceButton`. Communication with the Marketplace API is handled in the `markeplace.service.js`. A video walkthrough of implemented parts is included in the media folder of the enclosed CD.

As the comparison of the current master branch with the one containing this implementation is a long list, figure 3.3 is included just to provide some insight on the final scope of the implementation. Blue items are the ones only edited, while the green ones are newly added components.

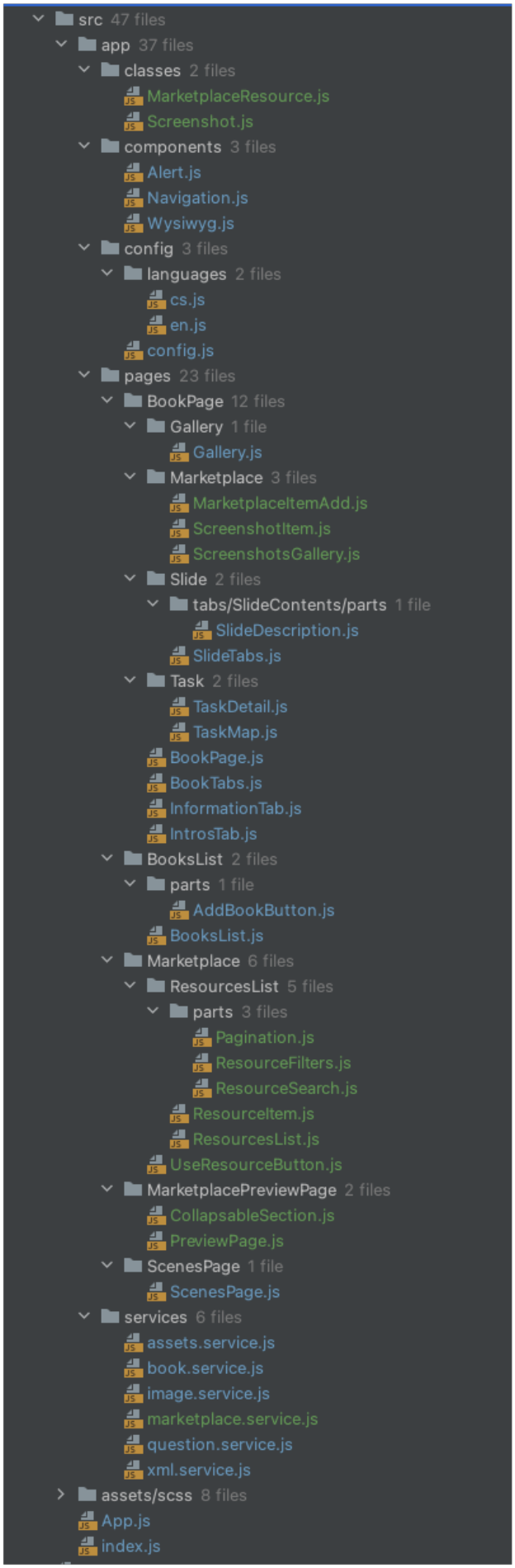

Figure 3.3: Comparison of a new source code structure with master branch.

The project is available on faculty gitlab[25] `https://gitlab.fel.cvut.cz/eduard/web_frontend.git` in the `marketplace_dev` branch. Its successful run relies on the parallel run of both implemented APIs.

3.2.1 Add resource to the Marketplace

`MarketplaceItemAdd` component handles the whole life cycle of the Marketplace resource. It initializes the item either as the first step after the click on the save button or if the user tries to upload the first screenshot, as the file upload requires an already initialized database entry for a reference id. Once the save button is clicked but the Marketplace resource has its switch set to unpublished, it sends a request to delete the entry as well as all associated files. When the publish switch is changed to "Yes" and the resource summary is empty, the component generates statistics about the current state of the book and generates the summary designed in section 2.2.3. The user is informed about the publishing (or unpublishing) of the resource via the `FloatingAlert` component. All mentioned features are depicted in the figure 3.4.

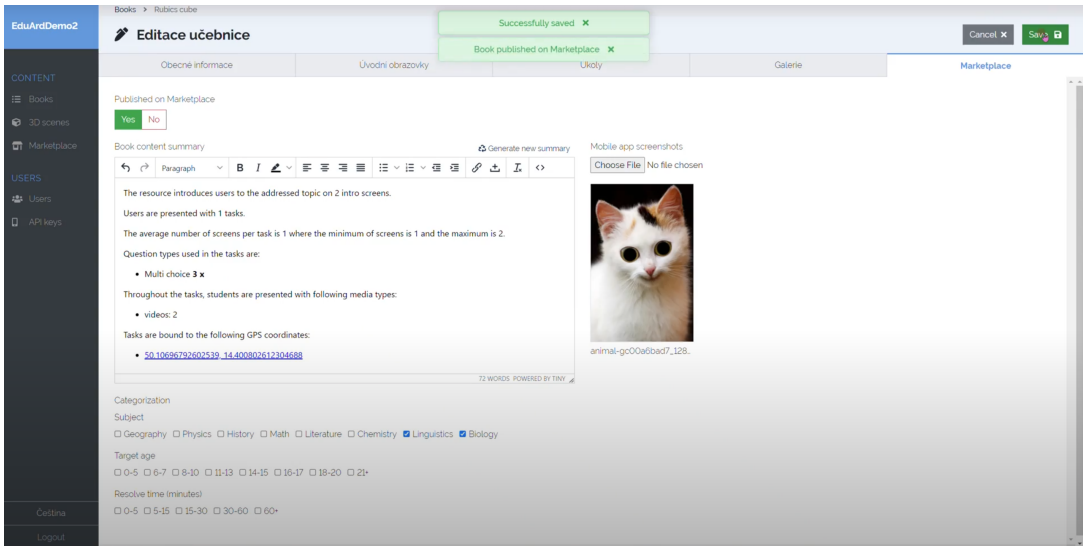

Figure 3.4: Publish resource to the Marketplace screenshot.

3.2.2 List resources on the Marketplace

`ResourcesList` component loads all resources available and the filter properties, groups all items into chunks of 12 for pagination and handles the filtering and sorting process as is shown in the figure 3.5. The same figure shows the `ResourceItem` component, which contains base information - thumbnail of the resource (taken from the thumbnail of the parent Book), title, short description and stats of the resource, based on which can the resources be sorted as well as the buttons to preview more information about the resource and to use selected one. Compared to the design, the categories of the resource are missing in this first version of the prototype.

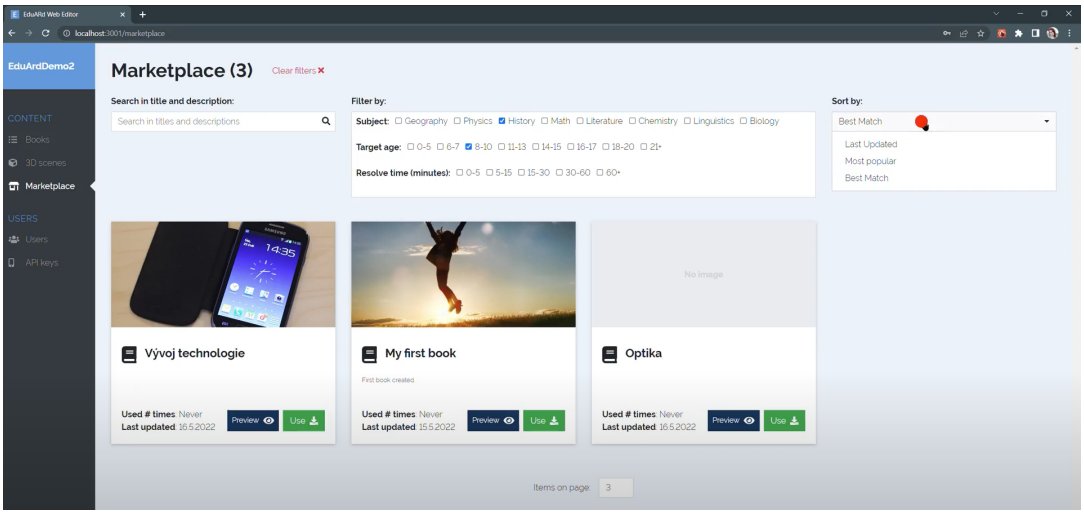

Figure 3.5: List, filter and sort resources on the Marketplace screenshot.

3.2.3 Show more information about the resource and use it

`PreviewPage` component follows the content and layout specified in section 2.2.4. In case the book content summary or description is not filled, the components are hidden. At the bottom of the page, the same `UseResourceButton` component as on the `ResouceItem` is presented. The button opens a modal window, with the prompt to insert a new resource title and after a successful addition, the user is informed by `FloatingAlert` followed by the redirect to the newly created Book. The process is shown in the figures 3.6 and 3.7.

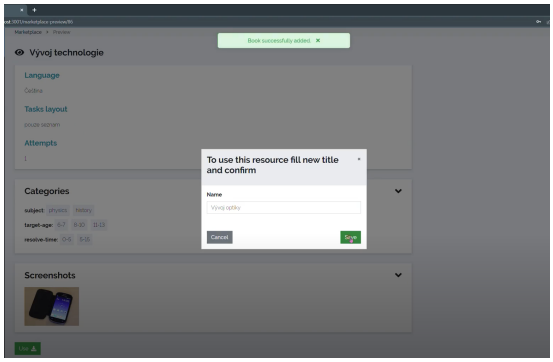

Figure 3.6: Prompt to fill a new title and save resource screenshot.

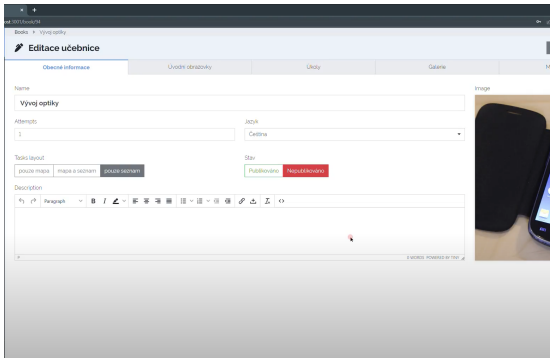

Figure 3.7: Redirected to a newly created book.

3.3 Future development

The features mentioned in the design chapter 2 and not implemented on the prototype would add to the overall user experience even more options and complexity. Before any of them is added, at least two rounds of qualitative user testing should be conducted, to confirm, that the additional features would help the users instead of adding the confusion or unwanted complexity.

List of features mentioned in the design, not included in the prototype:

- Claim the resource - section 2.2.3
- Show categories on the `ResourceItem` component
- Add resource as a variant - section 2.2.2
- Rate used resource and sort by rating - 2.2.2
- Expand filter properties and values

In addition to the listed ones, the results of the testing might define other points of confusion or provide insight into some other parts, where the user flow is not as easy and continuous as desired.

Chapter 4

Testing

This section designs two types of user testing to be applied to the prototype to support the UCD methodology of the project. Results from both types will lead to further enhancement of the design and the final application.

4.1 Qualitative testing - Usability study preparation

A usability study is the process of testing your application's design by observing users, measuring their performance, and documenting their comments[14]. This section creates two scenarios for such a study to allow the researcher to directly see the flaws and strengths of the application. The testing process, in addition to the specified scenario tasks, contains the introduction, acknowledgement, and outro after the session finishes, which follow the base polite behavioural principles. Proposed scenarios are tested on a small sample of participants and a few conclusions are drawn.

4.1.1 Scenario 1: Create Marketplace resource

The tasks of the Scenario 1:

1. Log in to the EduARd Web Editor.
2. Find a suitable book to publish to the Marketplace.
3. Fill in as much information as possible to show the potential users the whole content and scope of the book.
4. Save the Marketplace resource.
5. Preview created resource.

4.1.2 Scenario 2: Find suitable Marketplace resource

The following list of tasks tests the scenario structure with the complementing flow:

1. Write on paper the base content you would like to create.
2. Log in to the EduARd Web Editor.
3. Enter Marketplace.
4. Select suitable attributes to see the most relevant results .
5. Select the most suitable and use it.
6. Edit it further to fit your needs.

4.1.3 Testing process and conclusions

Two participants conduct both of the scenarios, each of them in a different order, in an attempt to exclude previous knowledge bias conclusions. Neither of them is a previous user of the EduARd system, so the base principles of the system are introduced to them as part of the introduction. Full recordings of both sessions are included in the media folder on the enclosed CD.

List of observations during the testing of the first participant:

- (Not)published on Book list is confused with publishing on the Marketplace.
- Selected filters are cleared when user goes to the preview and back.
- Missing message when no resource screenshots provided.
- GPS coordinates in generated summary on preview are a hyperlinks, but this fact is not visually enhanced.
- Overall experience rated positive, no confusion on how to complete assigned tasks.

List of observations during the testing of the second participant:

- Generated summary is obsolete.
- GPS coordinates are not informative enough, more textual info in the list would be appreciated.
- Selected filters are cleared when user goes to the preview and back.
- Selected filters are listing all resources that fulfill any of the selected values, exclusive approach would make more sense to the participant.

- Preview layout is too long and the most interesting values for the user are hidden under the fold.

- Overall experience rated positive, higher complexity would be appreciated.

Some of the observations resulted in the immediate fixes to the prototype - the missing message of the state without screenshots and missing visual interpretation of hyperlink on the Preview summary component. In addition, the option to generate the summary again is added in an attempt to prevent the problem with obsolete information on the generated summary.

As both of the participants are struggled by the clearing of the filters caused by previewing the resource, the first proposed change to the prototype is to change component `PreviewPage` to be always shown as a modal window. In that case, the filter state is preserved and the integration to the Marketplace tab will require no further implementation.

4.2 Quantitative testing - User questionnaire

The survey in the form of Google Form[26] is created and should be sent to the users who will interact with the prototype after a certain period to collect the insights. The questions are formulated using the Likert scale[14] to ensure the results will be measurable.

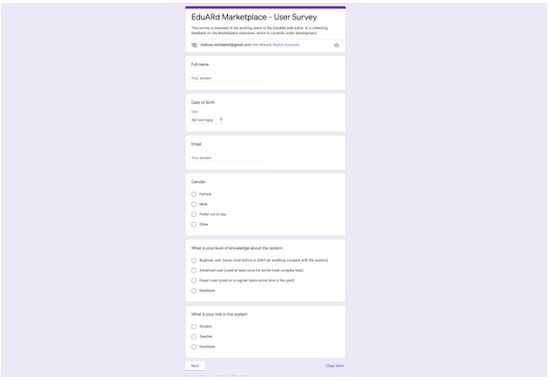

Figure 4.1: First step with the general information about the surveyed person.

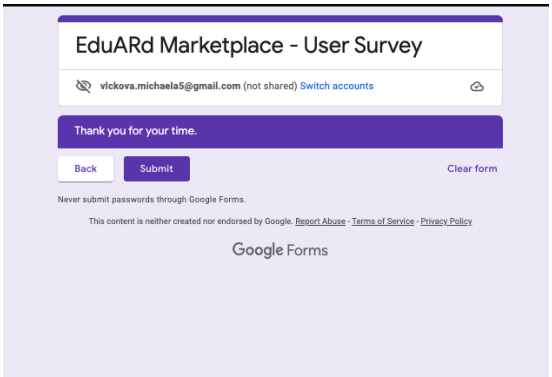

Figure 4.2: Survey: Last step with thank you message.

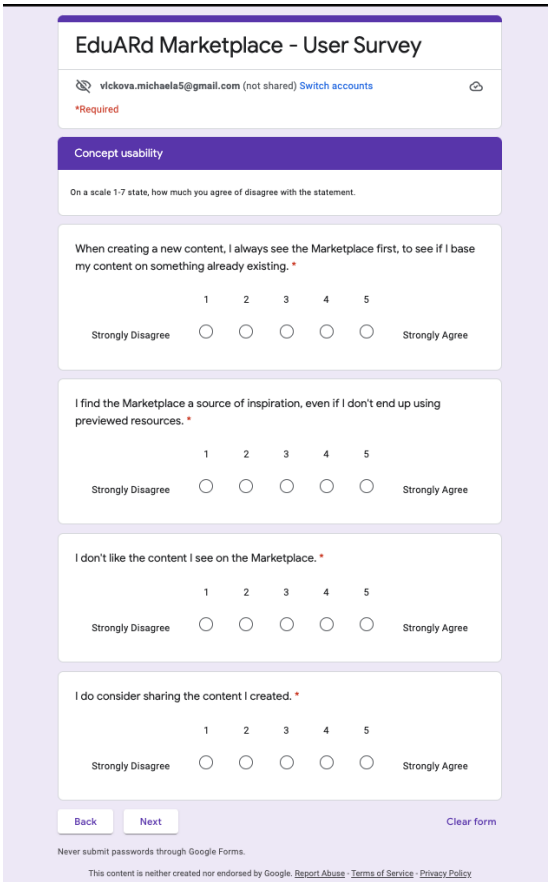

Figure 4.3: Questions about the overall concept.

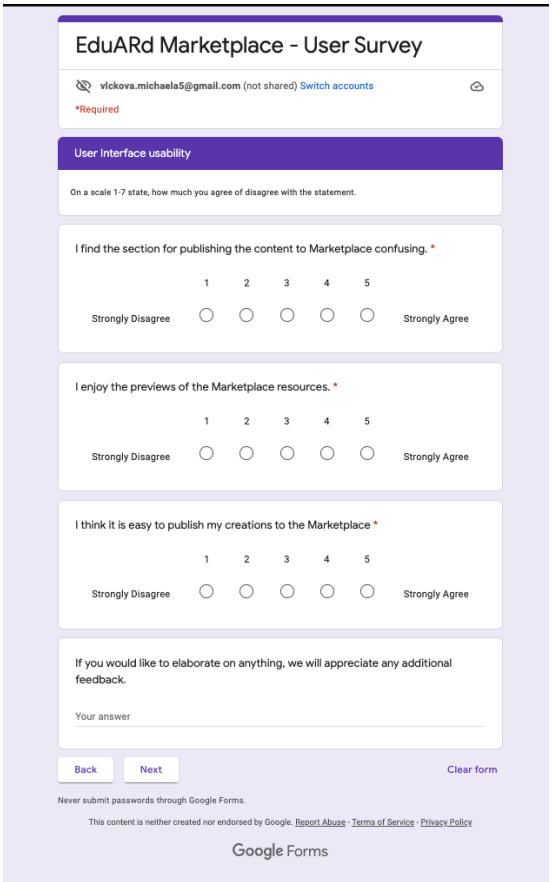

Figure 4.4: Questions about the implemented UI.

List of the questions: **Concept usability:**

- When creating a new content, I always see the Marketplace first, to see if I base my content on something already existing.
- I consider the Marketplace as a source of inspiration, even if I don't end up using previewed resources.
- I don't like the content I see on the Marketplace.
- I often consider sharing the content I created.

UI usability:

- I find the section for publishing the content to Marketplace confusing.
- I enjoy the previews of the Marketplace resources.
- I think it is easy to publish my creations to the Marketplace.

If you would like to elaborate on anything, we will appreciate any additional feedback.

It is expected that this type of testing will be repeated regularly to gather information on how much the users interact which the Marketplace, as well as if there might be new use cases from which they would benefit.

Conclusion

The EduARd project has a great potential to make the learning extremely interactive and full of rich experiences. The thesis expands the possibilities for teachers to share the created content, experiment with a lot more resources and share their creations.

The analysis is separated into the summary of the project's current state to pick the best approach for the extension development, the research of the user-centered principles to define the design process, and the research of the similar existing services.

The design chapter focuses on defining the user requirements, separating them into the use cases and stating the solutions for all of them. The second part of the design contains the definition of the architecture of the implementation and states that the API is created as a separate service and that the frontend part of development is integrated directly into the existing web editor. It also defines how the source files and assets are handled. All files are renamed and copied to a separate folder structure and then copied back to the web editor environment to ensure the reusability of the resources.

The implementation describes how the prototype implements the compulsory FR to satisfy UR: adding the resource to the shared space - Marketplace, listing, sorting and filtering of the resources. It also mentions all source code locations and describes its basic principles and structure.

The testing chapter focuses only on the user experience studies and proposes scenarios and surveys to collect feedback for further development. It then demonstrates the outcome of the small testing of proposed scenarios.

The result provides a foundation for sharing complicated structures as 3D models, scenarios or GPS bound educational paths while taking advantage of their structural representation.

Bibliography

1. *Eduard: Authoring výukových Aplikací s AR + GPS* [online]. ČVUT - Fakulta elektrotechnická, [n.d.] [visited on 2021-12-31]. Available from: `https://fel.cvut.cz/cz/vz/produkty/eduard`.
2. SKÁLA, Jan. *User rights management system for EduARd project*. 2019. BA thesis. CTU - FEL.
3. WADEPICKETT. *ASP.NET documentation* [online]. Microsoft Corporation, [n.d.] [visited on 2021-12-31]. Available from: `https://docs.microsoft.com/en-us/aspnet/core/`.
4. [Online]. PostgreSQL Global Development Group, [n.d.] [visited on 2021-12-31]. Available from: `https://www.postgresql.org/`.
5. *Swagger API Platform - Design and Document APIs* [online]. [N.d.] [visited on 2022-01-02]. Available from: `https://swagger.io/`.
6. SURIKOVA, Anastasia. *WEBOVÝ KLIENT PRO SPRÁVU MOBILNÍCH VÝUKOVÝCH ÚLOH SYSTÉMU EDUARD*. 2019. BA thesis. CTU - FEL.
7. MRÁZ, Michal. *Web Portal for Augmented Reality Preparation*. 2020. BA thesis. CTU - FEL.
8. *Java™ Programming Language* [online]. Oracle, [n.d.] [visited on 2021-12-31]. Available from: `https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html`.
9. *React native · learn once, write anywhere* [online]. [N.d.] [visited on 2021-12-31]. Available from: `https://reactnative.dev/`.
10. *App Store* [online]. Apple Corporation, [n.d.] [visited on 2021-12-31]. Available from: `https://www.apple.com/app-store/`.
11. *Google play* [online]. Google, [n.d.] [visited on 2021-12-31]. Available from: `https://play.google.com/`.
12. MAŇOUR, Šimon. *Paths of science - apps on Google Play* [online]. Google, [n.d.] [visited on 2021-12-31]. Available from: `https://play.google.com/store/apps/details?id=cz.ctu.fee.dcgi.eduard.sciencein.ukstezkyvedy`.
13. MAŇOUR, Šimon. *Virtual natural trails in React Native*. 2019. BA thesis. CTU - FEL.
14. *User-Centered Design: A Developer's Guide to Building User-Friendly Applications*. O'Reilly & Associates, 2013.

15. *Inbound marketing, sales, and service software* [online]. HubSpot, [n.d.] [visited on 2021-12-31]. Available from: https://www.hubspot.com/.

16. *The best assets for game making* [online]. Unity Asset Store, [n.d.] [visited on 2021-12-31]. Available from: https://assetstore.unity.com/.

17. *About Us* [online]. Sketchfab, [n.d.] [visited on 2021-12-31]. Available from: https://sketchfab.com/about.

18. [Online]. California State University, [n.d.] [visited on 2021-12-31]. Available from: https://www.merlot.org/merlot/.

19. *Balsamiq Cloud* [online]. Balsamiq Studios, LLC, [n.d.] [visited on 2022-01-02]. Available from: https://balsamiq.cloud/.

20. *Tinymce: React integration* [online]. TinyMCE, [n.d.] [visited on 2022-01-02]. Available from: https://www.tiny.cloud/docs/integrations/react/.

21. *Microservices Pattern: Microservice architecture pattern* [online]. microservices.io, [n.d.] [visited on 2022-01-02]. Available from: https://microservices.io/patterns/microservices.html.

22. *LINQ to DB: LINQ to DB (aka linq2db)* [online]. [N.d.] [visited on 2022-01-02]. Available from: https://linq2db.github.io/.

23. *Overview of Docker Compose* [online]. Docker Inc., [n.d.] [visited on 2022-01-02]. Available from: https://docs.docker.com/compose/.

24. [Online]. Oracle, [n.d.] [visited on 2022-01-02]. Available from: https://docs.oracle.com/cd/E14373_01/user.32/e13370/sql_rep.htm#AEUTL190.

25. *GitLab.com - GitLab DevOps Platform - Full DevOps Toolchain* [online]. [N.d.] [visited on 2022-01-02]. Available from: https://gitlab.com/.

26. *PhysicalFileProvider Class* [online]. Microsoft, [n.d.] [visited on 2022-01-02]. Available from: https://docs.microsoft.com/en-us/dotnet/api/microsoft.extensions.fileproviders.physicalfileprovider?view=dotnet-plat-ext-6.0.

27. *Rider: The Cross-Platform .NET IDE from JetBrains* [online]. JetBrains s.r.o., [n.d.] [visited on 2022-01-02]. Available from: https://www.jetbrains.com/rider/.

Appendix

A Content of the enclosed CD

- `README.txt` ..... the file with CD contents description
- `src` ..... the directory of source codes
  - `EduARdMarketplace` ..... directory with parts of the project
    - `frontend` . directory containing the whole frontend react application code
    - `backend` . directory with the code of the Marketplace backend application
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
- `media` ..... the media directory
  - `implementation` . screenshots and video walkthrough of the implementation
  - `testing` ..... user testing recordings