MASARYK
UNIVERSITY

FACULTY OF INFORMATICS

Dynamic Address Validation and Prefilling in Xperience by Kentico

Bachelor's Thesis

JAN MINÁŘ

Brno, Spring 2024

MASARYK UNIVERSITY

FACULTY OF INFORMATICS

Dynamic Address Validation and Prefilling in Xperience by Kentico

Bachelor's Thesis

JAN MINÁŘ

Advisor: Bruno Rossi, PhD

Department of Computer Systems and Communications

Brno, Spring 2024


FACULTAS ARTIS INFORMATICÆ
UNIVERSITAS MASARYKIANA

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

During the preparation of the thesis, I used AI tools Grammarly for grammar check and ChatGPT to improve my writing style. I declare that I used these tools in accordance with the principles of academic integrity. I checked the content and took full responsibility for it.

Jan Minář

Advisor: Bruno Rossi, PhD

Acknowledgements

I would like to thank my advisor Bruno Rossi, Ph.D. for his guidance, feedback, and support. Next, I would like to thank my consultant, Mgr. Marián Varaga, for valuable advice and consultations. I would also like to thank Ondřej Henek, MSc. for helping with the business aspects and Ing. Libor Mišurec with Mgr. Lucie Glaser for assisting with UX and introducing me to user testing. Finally, I would like to thank my family and friends for their support.

Abstract

With online forms becoming crucial to digital transactions and business operations relying on them, the importance of address validation and prefilling in those forms is often overlooked. Not implementing these features can have a significant impact not only on the company but also on its customers. This thesis aims to improve Xperience by Kentico by adding functionality to validate and prefill addresses. Available address validation APIs and their usage on other websites are researched and compared. The best-fitting one is selected and implemented as integration into Xperience by Kentico, focusing on making this integration user-friendly and dealing with problems arising from the support of many different countries. In the end, the usability of this integration is tested.

Keywords

Xperience by Kentico, DXP, .NET, Address Validation and Prefilling, Integration

Contents

List of Tables

List of Figures

Introduction

The last years have revolutionized how businesses operate, with online forms becoming crucial to digital transactions. These forms, from registration to checkout forms, often require users to input their addresses. However, the importance of address validation and prefilling in these online forms is usually overlooked despite its significant implications for businesses and customers.

Address validation involves checking a physical address against a database of known addresses to ensure its accuracy and deliverability. This process involves challenges such as handling variations in address formats across different countries, recognizing and correcting typographical errors in real time, and dealing with the ever-changing landscape of global addresses. Having accurate address information when shopping online ensures that the purchased goods or services are delivered to the intended location, which leads to higher customer satisfaction and reduces the likelihood of expensive delivery errors.

Address prefilling based on suggestions helps users fill in their addresses faster and correctly the first time. Adding prefilling functionality requires an intelligent system capable of predicting user input, understanding partial addresses, and offering the most relevant suggestions based on minimal user input. This system must be fast, accurate, and intuitive, leveraging complex algorithms and vast databases of address information.

Despite these benefits, many online forms do not incorporate address validation with prefilling or do so in a way that is not user-friendly. This lack can lead to user frustration, abandoned forms, and lost sales. Therefore, it is crucial to implement address validation in online forms in a way that is easy for users to navigate and understand.

This thesis aims to tackle these technical challenges by developing a solution for address validation and prefilling suitable for Xperience by Kentico, a product developed by Kentico Software s. r. o. The proposed solution encompasses the integration of address validation algorithms and a user-friendly prefilling system that anticipates user needs and enhances the overall user experience.

The address validation and prefilling concepts, together with application programming interfaces (APIs), most often providing validation and prefilling functionality, will be explained in the first chapter.

In the second chapter, a product, Xperience by Kentico, will be introduced with the company that created it.

The third chapter analyzes already-created solutions to this problem, what was done well, and what issues must be avoided. Then, this part compares available APIs, providing functionality for address validation and prefilling.

The fourth chapter focuses on developing the integration into Xperience by Kentico, which will provide the mentioned functionality using the selected API.

The last chapter will provide usability testing for this new integration, analyzing how user-friendly the solution is.

1 Address Validation and Prefilling API

Address validation and prefilling are essential features that each delivery company must consider when creating their website for ordering. This functionality is commonly supported by APIs. The following parts describe APIs, both the validation and prefilling functionality and show the process of integrating this functionality into the products.

1.1 About APIs

This section focuses on APIs, their types, and their architectures.

1.1.1 Definition of API

An API, short for application programming interface, is a set of rules that allows software applications to communicate with each other to share data and functionality. APIs streamline application development by enabling developers to integrate data and services from other applications. They provide a secure way for application owners to share data and functions within their organization or with business partners. APIs also enhance system security by sharing only necessary information. Each API should contain documentation to guide the developers using this API [1, 2].

APIs work by exchanging data through a request and response cycle. The API receives the request from a client, retrieves the data, and returns a response to the user.

API client

An API client is essentially a program, like a web browser, designed for use on a computer, smartphone, or other device. This software assists users in sending requests to desired APIs [3].

API request

An API request allows applications to exchange data or provides additional services [4]. Requests will vary depending on the type of API but typically include:

- Endpoint: A dedicated URL for accessing a specific resource.
- Method: Indication of the operation type on a resource using standard HTTP[1] methods.
- Parameters: Variables passed to an API endpoint for specific instructions.
- Request headers: Key-value pairs providing extra request details.
- Request body: Contains the data required to work with the resource.

API response

The API server sends a response to the client, typically including the status code, response headers, and response body. This data is usually in the form of JSON or XML documents [6].

1.1.2 Types of APIs

APIs can be classified in different ways, for example, by their accessibility level: Open, Internal, and Partner APIs.

Open APIs

Open APIs, also known as Public APIs, are designed to be accessible to any developer. These APIs are open-source and can be accessed using the HTTP protocol. Public APIs provide open access to an organization's data, functionality, or services, which third-party developers can integrate into their applications.

---

1. The Hypertext Transfer Protocol (HTTP) is crucial to the World Wide Web, allowing the retrieval of webpages through hypertext links. The protocol operates at the application layer, facilitating the transfer of information between networked devices [5].

Internal APIs

Internal APIs, also referred to as Private APIs, are developed for use within a specific organization and are not accessible to external users. These APIs enable communication among different development teams within the company by connecting various software components. For example, a social media platform might utilize multiple private APIs to handle various functions such as login processes, user feeds, and messaging systems. The primary goal of Internal APIs is to enhance internal development procedures and operational effectiveness.

Partner APIs

Partner APIs are specifically created to connect strategic business partners and facilitate the sharing of data or functionality to encourage collaboration on joint projects. Typically, access to these APIs is managed through a public API developer portal involving an onboarding process and authentication credentials. These APIs are not available to the general public and utilize secure authentication methods to ensure that only authorized partners can use them.

1.1.3 API architectures

APIs can be categorized based on the architecture they utilize. The most common architectures include REST, SOAP, GraphQL, and gRPC [7, 8].

REST

The most common architecture for transmitting data on the Internet is REST (representational state transfer). Within a RESTful framework, endpoints provide access to resources, and standard HTTP methods like GET, POST, PUT, and DELETE are used to perform operations on these resources.

SOAP

SOAP (Simple Object Access Protocol) is a lightweight XML-based messaging protocol that facilitates data transfer between client and server using various communication protocols. This protocol is commonly used in enterprise environments and legacy systems. SOAP may be slower than other API architectures, but this approach offers advanced security features.

GraphQL

GraphQL is an open-source query language and runtime for server-side applications, allowing clients to retrieve precisely the data they need from a single API endpoint instead of making multiple chained requests. This approach reduces the number of exchanges between client and server, which is advantageous for apps operating over slow or unreliable networks.

gRPC

gRPC, a modern RPC (Remote Procedure Call) framework, is an API created by Google that enables smooth communication between clients and servers in distributed applications. gRPC allows clients to invoke server procedures as if they were local, simplifying network communication.

1.2 Address Validation

Address validation, also known as address verification, is a process that ensures the existence and accuracy of street and postal addresses. This process involves checking an address against a database of known addresses to confirm its correctness and deliverability [9].

Address validation refers to a software solution that is utilized to authenticate customers' address information. This software is designed to identify and validate various components of an address. Validation's purpose is to assist businesses in saving time and money while also ensuring that they do not miss out on potential customers due to incorrect addresses. In some cases, these systems go beyond sim-

ply verifying the existence of an address and instead confirm whether the address is accurate and deliverable by comparing it to third-party reference data [10]. Moreover, these solutions also contribute to the smooth operation of a business by helping to prevent fraudulent activities, which could result in costly legal action and loss of revenue.

1.2.1 How Address Validation Works

Address validation can happen in two stages of the process, in which the user works with location-dependent services. The first is in the beginning, immediately when the user tries to submit his address into the related form. The second one is retrospective when the company references the submitted data with their address database after the order was submitted.

Address validation typically consists of three steps that ensure the correct handling of the submitted address. Those steps are called cleansing, supplementation, and standardization.

Cleansing

Cleansing refers to repairing information, like fixing typos, misspellings, and other errors. Address data may contain many errors, such as periods, commas, and odd numbers [11].

Supplementation

Supplementation is the addition of missing address components, such as a postal code. As users often fill in only parts of the address, it is essential to add any missing information so that the address is complete and can be easily used further [12].

Standardization

Standardization in the context of address management involves parsing, formatting, and normalizing addresses to ensure they are consistent and easily understood by postal services. This process includes placing the house number before or after the street name as per local conventions, converting abbreviations (e.g., "st." to "street") to their

complete forms, aligning address formats with postal service guidelines, and standardizing capitalization. The goal is to maintain all addresses in a standardized format to facilitate accurate and efficient further work with them [12].

All of the mentioned steps run on the server side of the API, focusing on address validation. The programmer using this API specifies how the validation should work and adds additional options for the validation. After the process is done, the programmer receives results that can be mapped into the code.

1.3 Address Prefilling

Address prefilling can be understood in many ways. In the context of this thesis, prefilling is a feature, sometimes known as autocomplete, that automatically suggests street addresses during user entry and fills them into respective form fields.

Address prefilling aims to minimize the need for typing, errors, and incomplete addresses during entry. Proper implementation of prefilling can streamline the checkout process, decrease cart abandonment rates, correct formatting to match local postal standards, confirm the existence and deliverability of addresses, and improve address data quality by preventing invalid entries.

Address prefilling offers numerous advantages for businesses and users alike. For businesses, prefilling helps maintain a clean and accurate database from the start, ensures the collection of deliverable addresses, prevents the collection of erroneously entered or intentionally fake addresses, and eliminates the need for frequent database cleaning. For users, this feature accelerates address entry, enhances address data quality by reducing errors, and reduces frustration stemming from incorrect address data.

The incorporation of address prefilling requires the utilization of an Address Autocomplete API. This API delivers real-time address predictions with each keystroke by receiving queries as the user inputs an address. API then matches searches for the input in the database of addresses and returns the most relevant results [13].

1.4 Integrating Address Validation and Prefilling

To create a successful system utilizing the functionality of address validation and prefilling, the company needs to choose the best-fitting solution for their use case and integrate the solution into their product.

1.4.1 Preparation

Many things must be considered before choosing the best-fitting solution to start integration development. These considerations consist of analyzing competitors' solutions, comparing all available services handling the problem, and writing down all requirements for the website with integration.

Analysis of Competitors' Solutions

Identifying your industry's competitors and studying their products is known as a competitor or competitive analysis. This information can be used to assess your company's strengths and weaknesses compared to each competitor and develop the best product possible. Researchers can conduct a competitor analysis at a broad level or focus on a specific aspect of your competitors' businesses [14].

Researching other address-validating or address-prefilling websites provides insights into how to approach this problem and helps identify possible usability issues that need to be avoided and good concepts that should be considered. Additionally, the comparison shows popular services behind the validation and prefilling that can be later compared to find the best-fitting one.

Requirements

Project requirements are the specific goals and features a project must meet to be successful. Requirements guide the project from start to finish, ensuring everyone works toward a common goal. Requirements often act as a roadmap for development, meeting stakeholder expectations within the set time and budget constraints [15].

The first big decision that needs to be addressed is whether the product should support both address validation and prefilling or

only one. The second one is the budget that will be used for this functionality, as services supporting address validation or prefilling have many different pricing plans. The next important thing is to take into account the technology in which the product runs and select the solution with the technology that will be the easiest to add. In most cases, address validation is supported using API, but some services also offer client libraries for specific programming languages that can make the developer's life easier. Address prefilling is usually run by API, JavaScript, or React components, so this also needs to be considered.

Comparing Available Services

Comparing available solutions is a common task for companies as they always aim to choose the best solution at the lowest price possible. This process involves evaluating multiple solutions, testing each one, and considering user feedback. Although it may seem time-consuming and costly, it is not, as the cost savings from selecting the best solution can be significant.

When comparing address validating and prefilling services, the first thing to validate is if the service fulfills the requirements previously defined. Second will be the coverage and reliability of the service, as there are significant differences between individual ones. Another consideration should be the quality of documentation and support for the service since poorly written documentation can lead to a much longer development phase.

1.4.2 Implementation

After the address validation and prefilling service is selected, the next step is integrating this service into the product. This section provides information on achieving both prefilling and validation using API.

Adding Suggestions

To enable address suggestions prefilling for address fields in forms, the first step needed is to create the field itself so that it will be rendered on the website. Secondly, the programmer must load the script

file containing a service that returns suggestions based on the input. After completing the previous steps, the programmer can activate the service once the user enters information into the address form, then display the selected suggestions in the address field [16].

Sending Validation Requests

When a user submits an address, this value must be sent to the API for validation. The more information is specified, the better the results. Selecting a country where the address is located improves the results the most. This improvement is caused by different countries having different address formats, which can cause issues when validating addresses worldwide [17].

Handling Validation Responses

After the validation request is sent, a response with the result is returned. This response contains information if the submitted address is valid and complete, and also formatted address. Each API returns The formatted address differently, meaning that creating one general feature that would work with all APIs is impossible [18].

One thing to consider when implementing this functionality is the possibility of the API not returning any response or returning an error. These scenarios need to be handled so that the application does not crash.

2 Xperience by Kentico

As this project will be integrated into the Xperience by Kentico, it is necessary to induce the product and company behind it. And since Xperience by Kentico is a digital experience platform (DXP), this type of system, as well as content management systems (CMS) that are a part of DXPs, will be introduced here.

2.1 Kentico

Kentico is a global company that provides an innovative Digital Experience Platform to businesses worldwide. Kentico has over 250 employees and a vast network of partners. The company is also committed to making the world a better place and takes responsibility towards its customers, partners, employees, their families, the environment, and wildlife [19].

Around 15 of the closest company's partners are called MVPs. They have a broad knowledge of Kentico and its products and participate in many companies' research projects. During the development of this integration, these MVPs will be participating in the research responding to the questionnaire prepared for them.

2.2 Content Management Systems

"A content management system is software that helps users create, manage, and modify content on a website without the need for technical knowledge." [20] As stated, content management systems have changed how content editors work. There is no longer a need for agencies to have a programmer who creates the code behind the final products, most often web pages. The code structure is prepared for them, and they have to add the desired content to its fitting places using an interactive user interface.

Every content management system should include some essential features. Firstly, content creation, which enables editors to input their own content easily. CMS should also offer content storage, maintaining all content in a single, consistent location. Additionally, workflow

management is crucial for assigning permissions and responsibilities among different roles. Finally, this system must have publishing capabilities, allowing users to push updates live.

There are mainly three types of content management systems, all shown in Table 2.1 and described bellow [21].

Table 2.1: Primary types of CMS

CMS type	Primary usage
Enterprise CMS	Internal control
Web CMS	Web Pages
Component CMS	Multi-channel

Enterprise Content Management Systems

ECMS systems provide a centralized platform where organizations can manage content related to their business processes. This content is easily accessible and helps employees with their work.

By allowing content to be stored in any file format, ECM systems provide unparalleled adaptability to meet diverse needs. They automate document management processes, enabling users to focus more on content creation and boosting productivity. Additionally, ECMs are budget-friendly, saving costs by ensuring that only essential data is stored while unnecessary data is efficiently deleted.

Web Content Management Systems

A WCMS is a type of CMS designed specifically for web content. These systems provide website authoring, collaboration, and administration tools that help content editors create and manage their websites without any programming knowledge.

By enabling the automatic publishing of information, WCMS significantly saves time and improves the efficiency of workflow management. This system also allows for personalizing webpage style and content, adapting to unique user preferences, and enhancing the visitor experience. Furthermore, WCMS's scalability ensures that as a business grows, the system can adjust without worrying about out-

growing website constraints, making WCMS an invaluable tool for companies aiming for expansion.

Component Content Management System

A CCMS is a type of CMS that manages content using components. These components allow for the reuse of content across multiple documents and outputs.

CCMS enables content re-usability, reduces translation expenses, and allows seamless distribution across various platforms. Additionally, CCMS provides traceability, enabling teams to track content changes, and enhances team collaboration by streamlining workflows.

2.3 Digital Experience Platform

A digital experience platform is a software framework that helps organizations consistently develop, manage, deliver, and optimize digital experiences throughout the customer lifecycle. DXPs aim to create, deploy, and continually improve websites, portals, mobile, and other digital experiences. They handle most data management and user experience so digital marketers can focus on customer needs and engagement [22].

DXPs include several components such as content management to create and distribute content across many channels, digital marketing tools to support analytics, SEO, and automation, e-commerce capabilities to manage product catalogs and process payments, personalization features to tailor content and experiences based on user data, and integration capabilities to enable seamless communication with other business applications like CRM and ERP systems.

Businesses that want to enhance their digital presence can benefit from DXPs in many ways. They ensure consistent digital experiences across all customer interactions, enable seamless integration with other business systems for efficient data and process flow, and offer scalability to handle large volumes of data and traffic. Moreover, DXPs are highly flexible, supporting various functionalities and allowing customization to cater to specific business requirements.

2.4 About product

Xperience by Kentico is a hybrid-headless digital experience platform combining content management and digital marketing. A hybrid-headless system supports headless channels and content sharing through GraphQL[1]. However, the product's main focus is to make content editors' lives easier by providing them with an easy-to-use visual interface in the Page Builder feature. Xperience by Kentico also allows editors to manage content across various websites, emails, digital channels, and devices, all from a single platform. Moreover, it unlocks the power of AI email writing and a broad set of marketing capabilities, such as personalization, contact management, forms, search, and SEO[2] support. The platform is customizable and can be integrated, offering SaaS[3] or on-premises deployment, as well as composability [26].

This product offers an administrative user interface page connected to each created project. The content editors log in and manage everything using the applications provided. The starting screen with the dashboard can be seen in Figure 2.1. There are many tiles, each belonging to a different application. For example, Content Hub stores content that can be reused across various web pages, emails, etc. Pages application allows editing of web pages using Page Builder, where the editor quickly fills in content into prepared fields. The Forms application makes creating forms simple using the interactive Form Builder feature.

1. "GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data." [23]
2. Search Engine Optimization helps direct and improves the quality of traffic to a website from a search engine [24].
3. "Software as a service (SaaS) allows users to connect to and use cloud-based apps over the Internet." [25]

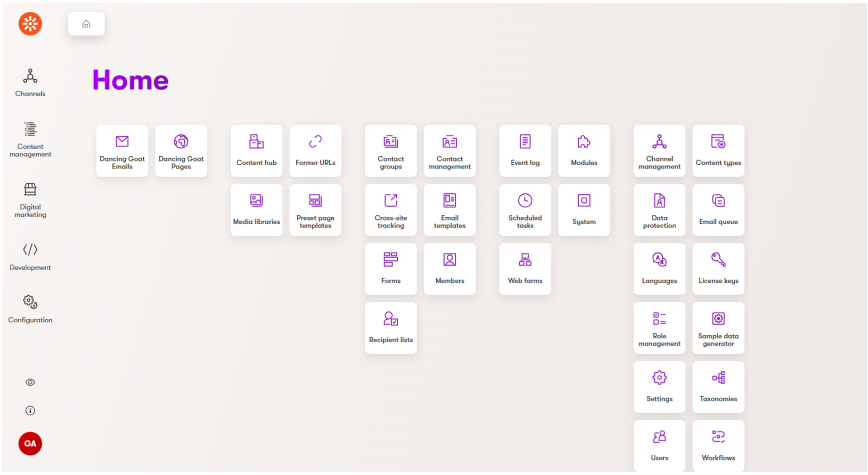

Figure 2.1: Xperience by Kentico Administration User Interface

2.4.1 Predecessors

Xperience by Kentico is not the first product created by the company. A significant change between the predecessors and the newest product led to a complete redesign. This also meant that all previously created integrations could not be used with Xperience by Kentico.

Its predecessors were called Kentico Xperience, with the addition of a number relating to the version of that product. The latest one from that series was Kentico Xperience 13, which is still supported now. These products were released once a year, with support remaining even after the release of the newer version.

Kentico Xperience was built on the ASP .NET framework using the MVC architecture. ASP .NET adds tools and libraries to the .NET framework for easier creation of web pages [27]. Meaning developers can create backend server and frontend views using this framework. MVC architecture tells how to design web applications. By following that pattern, they split their code into these three categories: Model (used for managing data and business logic), View (used for handling layout and displaying data), and Controller(used for routing commands to models and views) [28].

2.4.2 Integrations

If someone would like to expand the features available in Xperience by Kentico and make them publicly available, then integrations are the way to go. The product already offers a wide range of ready-to-use integration, and new ones are still being added. They are all available in their respective repositories under Kentico's GitHub account.

Programmers need to follow additional steps to create integrations, such as registering their services during application startup. However, Xperience by Kentico offers an API to assist them. Additionally, there are packages available that provide various classes and methods to simplify the integration process.

It is worth noting that every integration needs to go through a review process by Kentico to be accepted and promoted by them. There are two types of integrations: one with complete support and a 7-day bug fix policy and Kentico Labs, which are intended to inspire customers. There is no guaranteed support for them. Customers can create their bug fixes and submit them for review, as they could be later included in the integration itself.

3 Investigating Address Validation and Prefilling

Before starting the development of new features, it is crucial to begin with research to investigate all possible solutions and ensure optimal outcomes. This chapter delves into how users interact with existing features and identifies any aspects they might find challenging. By understanding and leveraging established patterns familiar to application users, designers can create more intuitive and practical designs. This ensures a smoother implementation process and enhances user satisfaction and engagement with the new feature [29, 30].

As the research covers a vast area, this project will focus only on some parts. First, the research project will be defined, and problems arising during the research will be identified. Next will be the study of other implementations to find standard practices used during address validation and mistakes that must be avoided. The following parts will be a questionnaire to company partners to get their insights on this topic and the research of available APIs handling address validation. Finally, a summary of the results obtained during this process will be created.

3.1 Defining Research Project

First was the definition of how this research should look like. This definition included filling in the Research Project Template our company's user experience (UX) team created for their research, and they shared this template so that it could be used in this project as well. This template outlines essential considerations for research, including the research problem, goal, expected outputs, research questions, methodology, and participants.

3.1.1 Research Problem

The research problem defines the situation the researcher is working with and tells him why it is essential to research this topic. The

definition should include a brief context, information missing at the beginning, and the benefits of dealing with this problem [31].

In this case, our customers could not easily add any validation of addresses to their forms. Hence, they either must rely on their users to submit valid data or implement this functionality independently. Implementing this functionality can be time-consuming, and relying on users is impossible because users always find some way to submit invalid data.

Missing information was in the following areas:

- How often do they create forms, and how important are they for them, where validation of addresses would make sense?
- Do they have any validation in their address fields as of now?
- Would they prefer faster local or slower worldwide validation?
- How does our competition handle address validation?
- What problems do users face when they are using validated forms?

Since address validation can be complex to implement and needs some third-party solution, not leaving this to our customers will make our product more usable when creating forms with address fields. Our customers could rely on the data these forms would send to them. Getting only valid addresses could be critical for businesses like delivery companies, where invalid data could lead to an inability to deliver products.

3.1.2 Research Goal

The research goal defines what is the primary goal of this research [32].

In this context, it means understanding how our customers work with forms containing address fields and finding out how they would like to handle validation and prefilling of addresses. Would they even find it helpful? Research other implementations that are available for validating and prefilling. Identify problems that users face when they are using validated forms.

3.1.3 Expected Outputs and the Use of Results

This part should mention how the results will be used after the research.

The results gathered here will be used in developing this integration's design and implementation phase. They will shed some light on handling and solving the address validation problem. Also, some key issues and usability problems that need to be avoided.

3.1.4 Research Questions

A research question tells what the researcher wants to find out in his research. Questions should focus on a single problem, be specific enough to answer thoroughly, and be relevant to the field [33].

- **RQ1**: How to handle submitted invalid data?
- **RQ2**: How many fields should the address be divided into?
- **RQ3**: How do other implementations handle validation?
- **RQ4**: How do other implementations handle prefilling?
- **RQ5**: How to add address prefilling based on company name?
- **RQ6**: Is it possible to merge the company name and address into one field?
- **RQ7**: What are the biggest obstacles for users when filling in addresses to the validated field?
- **RQ8**: Which address validation API best fits this project's needs?
- **RQ9**: How significant is this problem for our customers, and how often do they need to deal with it?

3.1.5 Methodology

Methodology outlines what type of methods will be used to achieve the research goal [34, 35].

The methodology used here included research of other implementations and a questionnaire for company partners.

3.1.6 Participants

Participants are people who are connected with the topic being researched, and information is gathered from them.

They will include company partners as they are the ones who will be working with the field implemented in this project when creating their forms. Additionally, they also have been direct users of those forms with address validation.

3.2 Research of Competitors Solutions

The next step was researching other implementations, finding good ideas, and avoiding mistakes. The research consisted of studying 9 websites that use address/company validation or prefilling in their forms. Since this sample should accurately represent worldwide use, the sample must contain websites of various types from various countries.

Table 3.1 compares features supported by researched websites. The first column focuses on address prefilling, the second on address validation, the third on address suggestions based on current location, and the last on company validation.

Table 3.1: Supported features on researched websites

Website	Prefilling	Validation	Current location	Company
Alza	✔	✔	✘	✘
Lieferando	✔	✔	✔	✘
Aliexpress	✔	✔	✘	✘
Rohlik	✔	✔	✔	✘
Air Bank	✔	✔	✘	✔
Red Hat	✔	✘	✘	✔
NAB	✔	✔	✘	✘
UPS Shipping	✔	✘	✘	✘
Popeyes	✔	✔	✔	✘

As can be seen, all of the researched websites support address prefilling, and the majority of them also address validation. However, when it comes to current location suggestions, only three websites support that, and only two support company validation, showing that

the later-mentioned features are not as important as the previous ones for these types of websites.

Table 3.2: APIs used on researched websites

Website	Used API
Alza	Smartform.cz API
Lieferando	Google Maps API
Aliexpress	Requests sent to their API
Rohlik	Mapy.cz API
Air Bank	Requests sent to their API
Red Hat	Demandbase API
NAB	Requests sent to their API
UPS Shipping	Requests sent to their API
Popeyes	Google Maps API

On the other hand, Table 3.2 compares APIs used in researched websites. Various APIs can be seen there, as only Google Maps API appears twice. Many websites send requests to their own API, validating addresses themselves or sending another request to a third-party API.

3.2.1 Alza

Alza is a Czech e-shop that sells mainly electronics, but as the shop grew, it now offers various goods, for example, drugstore, household items, or pet supplies.

The system's pros include its effective suggestion feature, which comprehensively covers all addresses in the Czech Republic, and the requirement for including a house number in the address, which is logical for delivery purposes.

However, there are several drawbacks. The process of entering the city is inconveniently placed on a different page from the rest of the address details. Additionally, the system allows invalid addresses to pass through validation, and the validation of street names does not consider the city information provided earlier. The field also permits searches for city names, implying that this field searches for entire

addresses, including city and zip code, but the field only displays the street name and house number. Furthermore, the system is limited to Czech addresses only.

In summary, while the suggestion feature works well for Czech addresses, the system introduces unnecessary steps, such as navigating between two pages to adjust the city, which prolongs the process more than desired.

3.2.2 Lieferando

Lieferando is a German food delivery company also known as Just Eat Takeaway worldwide.

On the positive side, Lieferando's validation and suggestion features work well, and its current location suggestions make submitting addresses easier. This system supports only German addresses, which makes sense as Lieferando operates only there.

However, there are areas where this website could improve. For instance, the platform sometimes requires many characters before suggesting options, slowing down the address-submitting process. Additionally, the validation feature is only available on the initial page, not the subsequent page where users fill in their order details. The division into two pages seems unreasonable, especially when only the first page offers validation.

3.2.3 Aliexpress

Aliexpress is a Chinese e-commerce platform that offers a wide range of products at very low prices, which often indicates a compromise on the quality of the products.

One of the pros of this platform includes suggestions and validation as a quality-of-life improvement, with the interface adapting suggested addresses based on the country selected.

On the other hand, there are notable drawbacks. The platform struggles to recognize some streets in larger towns and is often unaware of smaller villages. This lack of comprehensive address knowledge makes the API somewhat ineffective. Furthermore, there is no validation when the "Other" option for cities is selected.

The attempt to solve the issue with bad API coverage by adding an "Other" option or removing validation in specific fields, paradoxically destroys the entire validation process, as users can easily bypass validation, leading to potential problems in address verification and delivery accuracy.

3.2.4 Rohlik

Rohlik is a Czech e-shop that gained most of its popularity during COVID-19. Rohlik primarily offers food as other supermarkets do but also has drugs, household tools, and much more.

One of the notable features of Rohlik is its interactive map, which enhances the user experience. The system provides good validation and suggestions coverage for address inputs and current location suggestions.

Unfortunately, the platform is not without its problems. Currently, there is no option to send an address that is not supported by the system, which can be limiting for some users. Additionally, searching for an address on the map can be buggy, leading to potential frustrations.

In summary, Rohlik presents a complex yet well-integrated solution for online shopping. This solution ensures that validation and prefilling are consistently applied across all fields related to addresses, leaving no room for bypassing validation. While a limitation, the inability to input unsupported addresses contributes to making the solution safer and more robust.

3.2.5 Air Bank

Air Bank is a Czech bank that provides solutions for managing various banking problems online.

Their website finds a good middle ground between strict and no validation, compelling users to double-check potentially invalid data while offering good suggestions and validation coverage.

However, there is one significant limitation to their services. The validation for companies and addresses is limited only to the Czech Republic. This limitation can be problematic, as users in other countries cannot open bank accounts there.

3.2.6 Red Hat

Red Hat is a company with a global presence, primarily offering products related to the Linux operating system.

Their solution provides suggestions for worldwide company names and includes most of the reasonably large companies. This approach appears to be one of the few examples where an API suggests or validates a company name. In contrast, most other websites do not validate company names at all. Typically, companies needing this information to initiate cooperation with other businesses allow them to fill in a form without validation and then get back to them individually.

There are also negatives with this website: it lacks any form of validation, does not provide information about the validity of the submitted address, and the selection of a country does not influence the companies displayed.

3.2.7 NAB (National Australia Bank)

NAB is recognized as one of the four largest banks in Australia.

NAB's website offers the possibility to manually fill in an address without validation if the address was declined by the system, which can be particularly useful when the system does not accept the address automatically. This feature is an uncommon practice in the banking sector.

The platform has a few problems, including validation restricted to only one field containing part of the address, support exclusively for Australia, and a lack of recognition for smaller villages. Additionally, the website includes two buttons or links that unveil extra fields, leading to a large form that users must fill in.

3.2.8 UPS Shipping

UPS, one of the biggest delivery companies in the USA, offers its address validation API for public use.

The pros of using this website include that addresses are suggested only to the relevant country, the system supports validation in all countries listed in the selector, and has good coverage, suggesting locations even in small villages.

However, there are some cons to consider. For example, no suggestions are offered in the second step if the user selects to fill in the address by its parts. Additionally, the API has no complex validation, only requiring users to fill in some value but not checking if the value is a valid address. This problem can lead to the UPS company ending up with an invalid address despite the API's good coverage worldwide and its effective suggestions system. And finally, there is no clear explanation of what to fill in fields labeled: Address Lines 1, 2, 3, etc.

3.2.9 Popeyes

Popeyes is an American fast food chain known for delivering its products directly to customers' homes.

The system has good suggestions and validation coverage, boasting a simple yet robust design. The main problem with this website is that it lacks a "Check mark" to indicate when a valid address is successfully added, which could ruin the user experience.

Despite this, the solution remains straightforward and effective. While this web may not feature some of the user experience enhancements seen on other web pages, Popeyes' website effectively prevents the submission of invalid data by allowing orders only through selected suggested addresses. Overall, this approach gets the job done efficiently.

3.3 Research Questionnaire

Since research questions were already defined, the next step was transforming them into questions in the research questionnaire. Those new questions should be easy for respondents to understand, and respondents should not need much time to answer those questions [36].

This questionnaire consisted of 13 questions, some helpful in obtaining context about respondents. The questionnaire was sent to almost 10 company partners (MVPs) , with whom Kentico cooperates the most. In the end, there were 8 responses with an average completion time of 9 minutes.

All of the responses gave valuable insights into how the partners would prefer this integration to work. Some respondents also provided suggestions for APIs they have used, expressing satisfaction with their reliability and supported features. After the evaluation of responses, most of the research questions could have been answered.

A complete questionnaire with all questions, the reasoning behind them, and responses can be found in the Appendix section Research Questionnaire.

3.4 Research of Third-party API Solutions

After researching other implementations and getting valuable knowledge from partners' responses, the next step was to study available APIs offering address validation and prefilling.

There were a few requirements for the API to be selected. The one selected should, of course, work reliably and have good coverage. Also, the more countries supported by this API, the better. The next one was that the API should offer at least one free pricing tier so that it is easy for customers to try that API out and implement it into their products.

With those requirements in mind, the research started with APIs seen in previous research of other implementations, followed by the most commonly used ones, and ended with suggestions from the questionnaire.

Table 3.3: Supported countries and functionality in researched APIs

API	Countries	Functionality
Smartform	Czech Republic Slovakia	Address validation & prefilling Company name validation & prefilling Search based on coordinates
Google Maps	Major ones	Address validation & prefilling Places search Search based on coordinates Current location search
Mapy.cz	Major ones	Address validation & prefilling
Mapbox	Major ones	Address validation & prefilling
Geoapify	Major ones	Address validation & prefilling
Smarty	Major ones	Address validation & prefilling
Loqate	Major ones	Address validation & prefilling Company names suggestions

Table 3.3 shows APIs with the countries supported. With one exception, all of them support most of the major countries. Next, there is a column with supported functionality that shows the capabilities of a given API. As can be seen, all of the APIs support address validation and prefilling, with some having additional features.

Table 3.4: Presence of free plan in researched APIs

API	Free plan
Smartform	✘
Google Maps	✔
Mapy.cz	✔
Mapbox	✔
Geoapify	✔
Smarty	✘
Loqate	✘

Since one of the requirements for API was having a free tier, Table 3.4 compares researched APIs with this criteria. The majority of them support the free version, but some do not and are therefore not suitable for this project.

Smartform

Smartform offers two possible solutions. One is Smartform for Webs, which provides a predefined JavaScript component ready for use. The second one is Smartform API, which offers traditional API responding with data to requests sent. Coverage on this API is good and also finds small villages. However, there is no free plan.

Google

This API is one of the most used and complex ones that allows even more features than stated here, but they will not be most likely helpful for this project, so they are not mentioned. The coverage of the API is good in most supported countries. The user of this API receives each month free USD 200, which he can use. After going over this limit, he has to pay for the additional requests.

Mapy.cz

This API offers good coverage in the Czech Republic and worldwide. Every month, the user receives free credits that he can use. He pays after exceeding this limit. This API also supports open-source projects with better packages that have more credits for free price.

Mapbox

Mapbox offers an easy-to-use component as an alternative to using their API. The big problem with the API is that address prefilling works poorly for Czech addresses in their online demo, which is concerning at best. They have good coverage and can find most of the inputs but cannot handle street names and house numbers together. When it comes to pricing, Mapbox offers one free tier, but to use this API in production, paying for a better tier is necessary.

Geoapify

Geoapify offers excellent coverage, using OpenStreetMap to get data. Geoapify has one free plan that could be useful for testing purposes, but afterward, this API needs to be paid to work in production.

Smarty

Smarty is one of the most used APIs in the world. This API has a fast request speed and supports a large number of countries. However, Smarty comes with a hefty price tag and no free plan, only a monthly trial.

Loqate

This API has no free plan, only a free 14-day starting trial, which could be complicated for testing purposes. In the demo, this API worked only for localized countries, which could also be a problem if it worked like this in production. But coverage was good in this one country.

3.5 Research Results

After completing the investigation, the data was collected and summarized to serve as a reference for the design and implementation phase. The sources of data included not only research on implementations, APIs, and responses from a questionnaire but also discussions with colleagues from the company. These colleagues have significant knowledge of the related topic and often interact with company partners and other customers using Xperience by Kentico.

3.5.1 Answering research questions

How to handle submitted invalid data?

The questionnaire showed various responses from the partners, but they all agreed in their disapproval of the "reject invalid submissions" approach. Most respondents preferred either accepting the data even if it had not been validated by the API or displaying an indication to the user that the data was not validated and allowing them to decide how to proceed.

In the end, that issue will be resolved by making validation optional. Content editors at agencies using Xperience by Kentico will have the option to decide whether or not they want to enable validation in the settings of the newly added field. As it stands, most of them rely solely on the suggestions provided by the API in some of their

projects, bypassing validation entirely. Additionally, they will have the possibility to add an extra field to the submission form, allowing users to specify their address more accurately should the address fail the initial validation.

How many fields should the address be divided into?

The discussion about this question involved a UX researcher from Kentico and a product evangelist who has extensive knowledge of both Xperience by Kentico and its application in customer projects. Additionally, web pages such as one from Adam Smith [37] were referred to in order to find a solution.

It was determined that although both approaches (single field and multiple fields) are used by customers, the single field approach is more commonly used. Moreover, it is easier for users to fill in data when presented with only one field rather than two or more. Therefore, only one field will be present in the final design. This decision will also make implementation more straightforward, as adding support for address validation into several fields could create problems with the correct display of suggestions, making it hard for users to orient themselves.

How do other implementations handle validation?

This topic was discussed in the previous section, Research of Competitors Solutions.

How do other implementations handle prefilling?

This topic was discussed in the previous section, Research of Competitors Solutions.

How to add address prefilling based on company name?

This feature is supported by some of the researched APIs that allow sending the company's name to it and returning its address. When implementing the feature, only enabling the API and correctly handling returned data is needed.

Is it possible to merge the company name and address into one field?

It is possible to merge the suggestions for addresses and company names in one field using certain APIs. Allowing also for the correct display of the address instead of the company name in the submission overview if it was entered. Validation will also function correctly when a new request is sent to convert the company name into an address.

What are the biggest obstacles for users when filling in addresses to the validated field?

Many problems can be seen in Table A.2. Some feedback about problems was also collected from colleagues, who usually mentioned one or more of the problems already present in that table.

Which API will be the best fitting for our needs?

This question will be answered in the section Chosen API.

How significant is this problem for our customers, and how often do they need to deal with it?

Based on the responses in the questionnaire, our customers frequently encounter issues with address submissions in their fields. Additionally, they do not often gather company names but would appreciate it if this feature was included in Xperience by Kentico. Finally, the customers have expressed keen interest in this functionality, as evidenced by most of them leaving their email addresses in the questionnaire to test the integration out before it becomes publicly available.

3.5.2 Chosen API

After conducting research on available APIs and defining the necessary requirements, one API stood out as the best fit: the Google Maps API.

The Google Maps API offers excellent coverage, with support for all major countries in the world, as documented on their website [38]. Additionally, the API includes a free tier with limited resources,

allowing customers to try out the feature before committing to a paid plan.

The Google API offers many useful features, such as pre-filling forms based on company names and retrieving the current location of a device, making the API a valuable resource for this project. Furthermore, because Google Maps is a well-known platform, it will be easy for customers to integrate this integration into their products and for their users to fill in data using this API.

4 Development of integration

After the research was done and the suitable API was chosen, the next step was implementing this integration. This step consisted of four main parts: designing how the integration should work, choosing technologies that will be used, implementing this feature as integration, and testing the newly created feature.

4.1 Design

Design is the essential part of creating products that are user-friendly and easy to use. This part focuses on how the users feel and how easy it is to complete their desired tasks with designed products. The design phase often includes the research that was already discussed in chapter Investigating Address Validation and Prefilling.

During the design phase, the 4D methodology was used that works with the 4D model that will be introduced [39]. After that, the primary design considerations regarding options for the newly created address field raised during this process will be discussed.

4.1.1 4D Model

The 4D model is an iterative process used in software creation's design and development phase. This model is centered around users and has four phases: discover, define, develop, and deliver. Figure 4.1 shows all mentioned phases and two diamonds that gave this model the alternative name of the Double Diamond Model.

The use of diamonds serves as a metaphor for the designer's creative process during the design phase of product development. At the outset, the designer begins with a blank slate and conducts thorough research to gather relevant information. Once the designer reaches the quarter mark of the model, he evaluates all the ideas generated so far and eliminates those that are not suitable. The process then repeats, with the designer generating possible solutions and selecting the best-fitting one.

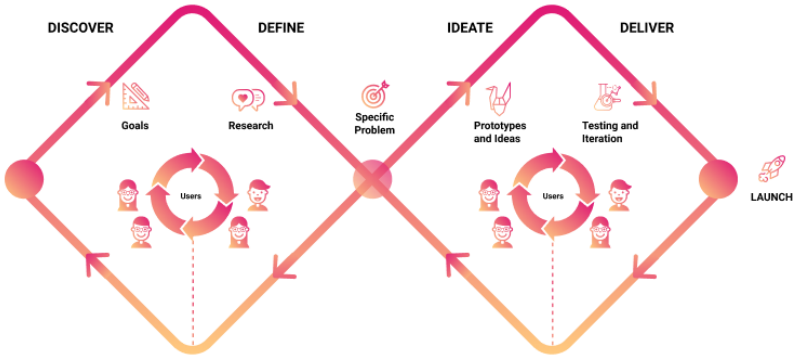

Figure 4.1: 4D Model used in Design phase [40]

Discovery Phase

In the discovery phase of a design process, extensive research is conducted to deeply understand the target users by identifying their challenges and observing their work environment to gather insights. This phase encompasses defining the project's requirements and specifications to ensure clarity before moving forward.

Define Phase

In the define phase of the design process, designers define the problem they face. Problem definition involves patterns finding and reducing the area of focus so that the following phases work only with relevant ideas.

Development Phase

The development phase of the process generates possible solutions to the problem. No solution is wrong in this phase, as it could lead to important discoveries. This phase is often iterated together with the following one.

Delivery Phase

The last phase of the project design process involves creating the initial prototypes, eliminating unsuitable solutions, and identifying potential implementation issues. New findings can arise at any point during this phase, which may require starting the process from the beginning. The final part of this phase involves developing a production-ready solution.

4.1.2 Design Considerations

During the design phase of this project, most of the questions that came up were regarding the possible options for how the content editors could set up this new address field in the form. All of them were thought through with all the problems they may introduce and the benefits they would give to editors. In the end, all decisions made were approved by the company's UX designer.

Table 4.1: Address fields options settings

Possible option	Present in the final design
Switch to enable address suggestions	✔
Switch to enable address validation	✔
Switch to enable company name submissions	✔
Switch for the position of the suggestions panel	✘
Switch to change the number of fields	✘
Input for countries to be supported	✔
Switch to enable current location suggestions	✔

Possible options are shown in Table 4.1, with only two not being implemented in the final version. All other options were considered beneficial for content editors using this integration and will be present in the final implementation.

The decision to not include a switch for the position of the suggestion panel (above or below the address field) was made because it would not be practical. Users often change the position of the address field by scrolling, so setting a fixed position for the suggestion panel during the creation of the form would not necessarily result in better rendering on their screen.

The decision against creating a switch for a number of fields was made after research showed that one field is the preferred option by users. This was also supported by knowing that implementing multi-field addresses would not allow communication between all fields in Xperience by Kentico, resulting in a bad user experience with suggestions and validation of addresses.

Other areas of design did not reveal any problems or questions. The visual design of the field itself is the same as for all different fields present in Xperience by Kentico, and for the suggestions panel, it was decided that the panel should look simple and display the logo of Google, not to break any terms of use. Developers of agencies using this integration will be able to modify this panel's look to match the design of their web pages.

4.2 Used Technologies

.NET

As Xperience by Kentico runs on a .NET framework on the backend, this framework is also used to create integrations that do not extend the current administration UI as that is written in React. This created integration used the latest .NET 8 for most of the code behind it. The backend code is written in c#, and the frontend field's look and functionality are in the MVC View file.

Javascript and CSS

To achieve the address suggestions, this integration uses JavaScript inside the view file and imports Google's JavaScript library containing service for autocomplete requests. CSS inside the View file is used for the suggestions panel's style.

4.3 Implementation

During the integration implementation, four tasks needed to be completed. First, the startup configurations had to be created, followed by the development of a basic address form component. The next

step was to add address validation, and finally, address prefilling was implemented.

4.3.1 Startup Configurations

A startup extension method needs to be created to ensure that this integration can be used inside another Xperience by Kentico instance. This method is called inside the customer's *Program.cs* file and gets the API key used for requests from the *appsettings.json* file. Knowing this key, the service sets up *GoogleMapsOptions* class that will contain this key and could be later used in other classes. Next follows setting up *HttpClient* using factory pattern[1] so programmers can use it using *HttpClientFactory* class. The final part of this extension method is registering all service classes and their interfaces so they can be later used by dependency injection[2].

4.3.2 Address Form Component

In Xperience by Kentico, adding new custom fields in forms is possible by creating new Form Components. To create a new Form Component class, the programmer needs to inherit from the *FormComponent<TProperties>* class available in Xperiecne by Kentico's code and override this component's *Value* getting and setting as well as *Validate* methods. *Value* property is used to store input from the user and later is displayed on the submission's page. *Validate* method is called after the form submission, and the programmer can add his custom validation logic here. This method returns a list of validation errors, which are then displayed below the respective fields in the submission form.

FormComponent for proper functioning also needs properties class implementing *TProperties* interface. This class adds form component options settings for content editors in the administration UI. All of these options are stored in the properties that are registered using the UI Form Component Attributes.

---

1. The Factory pattern centralizes the creation of objects in a single location. This approach uses factory methods to generate objects without specifying the precise class [41].
2. An approach called dependency injection used in programming involves providing an object or function with other objects or functions it needs instead of generating them internally. [42]

The final steps are to register the newly created component using the [*assembly: RegisterFormComponent(...)*] attribute and create a partial View file that will render this component [43]. In the View file, the HTML input should be mapped to the *Value* property defined in the *FormComponent* class.

4.3.3 Address Validation

Two new services were created to add support for address validation. Both services are called inside the *Validate* method of the address form component.

First was *AddressValidator* service, which calls Google's address validation API. This service has one public method called *Validate* that takes the inputted address and country restrictions set in the properties class. Both of those values are a part of the API request. The response is mapped to the created *AddressValidationResponse* class. The service returns the formatted address if the inputted address is valid and logs all errors caused by the API to the Event Log application.

The second service created is *AddressGeocoder* service calling Google's geocoding API. This API can transform company names into addresses that belong to them. Geocoder service has one public method called *Geocode* that takes the inputted value and country restrictions just like the *AddressValidator* service. The service returns a formatted address or logs any error similar to the previous service.

Both service calls from *Validate* method can be skipped during form submission using options in the administration UI.

4.3.4 Address Prefilling

JavaScript and CSS files were added with rendering services to enable address autocomplete.

First was created *GoogleMapsScriptsRenderer* class with two public methods. One method returns a script with necessary functionality created in Google's autocomplete API. The second returns JavaScript and CSS files that use the previous script and map the autocomplete API's functionality to the address field.

The JavaScript file contains one method called *initializeAutocomplete* that contains multiple nested functions. Each is responsible for one

task, like creating an item in the suggestions panel, showing or hiding the panel, and creating a button with a current location suggestion. The autocomplete service exposed by Google is initialized on each page load, and requests are sent to this service with the input the user has typed in. Suggestions returned are mapped to the items in the panel.

The whole suggestions panel had to be created from the start as the one offered by Google does not support the current location button, and adding a new custom one is impossible. The panel is created with minimal styling, as a developer will modify the panel's look later to match the final website design. To alter the panel, all CSS classes are listed in the *ReadMe* file with a short description.

To not break Google's usage policies, Google's logo is displayed at the bottom of the suggestions panel.

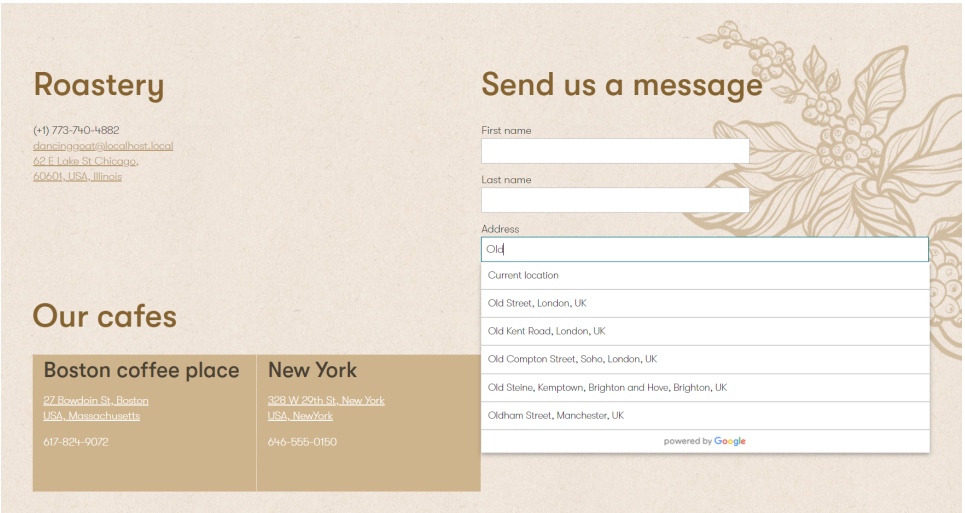

Figure 4.2: Sample Dancing Goat website with added Address Form Component

Next was *GoogleMapsTagHelper* class that uses tag helper[3] approach to get scripts returned from *GoogleMapsScriptsRenderer* to the customer's website layout.

3. ASP.NET Core Tag Helper is a feature that enables developers to add dynamic attributes to HTML elements within an ASP.NET Core MVC application [44].

Finally, the View file was extended with a call to the method in the JavaScript file that enables the autocomplete functionality. As this method requires several parameters to work as expected, those parameters are gathered in the View file from the address form component properties and transformed to values supported in JavaScript. The method call is done after the whole page is loaded thanks to *DOMContentLoaded* event.

4.4 Testing

All of the services and extension methods in this integration are covered with unit tests. These tests ensure that basic functionality in related code works as expected. They, however, do not check API's functionality, as it would not make sense to check code developed by someone else. This means that all requests sent by services are mocked[4] with predefined responses.

To ensure everything works as expected, manual testing was done with scenarios that users can achieve using this integration. Testing was done on *DaningGoat* sample project that is available with Xperience by Kentico. A new submission form was created with an address field present inside. The following scenarios were tested (the last five with combinations of address validation and company name support enabled/disabled):

- After adding the address field in the Forms application, the field is correctly rendered on the website.
- Enabling/disabling all of the options for the address field changed the field's behavior correctly.
- Adding country restrictions and suggestions language options correctly changed suggestions and validation.
- User clicks on the current location button with enabled/disabled browser access to the current location.
- User submits valid address.

4. Mocking is a software testing technique where a mock object is created to simulate the behavior of the original object that is temporarily replaced. [45].

- User submits invalid address.
- User submits only partial address.
- User submits valid company name.
- User submits invalid company name.

4.5 Installation and Setup

Installation and setup of this integration can be done in the four following steps:

1. Add the necessary configuration to the *appsettings.json* file.
2. Add provided startup method to *Program.cs* file.
3. Add the new import to the *_ViewImports.cshtml* file.
4. Place the Tag Helper to the *_Layout.cshtml* file.

The complete installation instructions can be found in Appendix parts Integration Installation and Google Cloud Setup.

4.6 Future improvements

Although this integration meets all the requirements that were set at the beginning, there is still room for improvements that can be made in the future. Areas that can be improved are the following:

4.6.1 Official support

Currently, this integration is available in the form of a project that can be added via reference to the target Xperience by Kentico project. Even though this approach is working, adding projects via reference is not the most practical approach, as programmers could end up with large folders when adding multiple projects like this.

This problem can be solved by making this integration official and publishing a repository containing this integration under Kentico's GitHub account. All public repositories there are also published as NuGet packages that are easy to add to already created projects. Before

the integration can be published, a review by Kentico's programmer needs to be completed together with a complex testing process by an experienced tester and a documentation review.

4.6.2 Multiple input fields

The research showed that one input field for an address is more user-friendly and easier to use than multiple fields containing parts of an address. However, there are still companies that prefer the second approach. This approach is not currently possible as the address component is rendered only as one input field. Adding this option to select input fields' count would require modifying the code responsible for rendering the component as well as code handling API requests and responses. Adding multiple fields for addresses could possibly lead to usability problems as countries have different formats of addresses.

5 Usability Testing

This chapter focuses on testing the usability of the integration created. First, information about user testing will be provided. The next part will focus on steps during testing and tested scenarios, and the final part will evaluate the results obtained.

5.1 About Usability Testing

Usability testing is a method used to evaluate how easy to use and effective a product or system is by involving real users or individuals with significant knowledge about the product in its exploration. The primary aim of this testing is to uncover any usability issues, gather feedback, and measure the users' satisfaction with the product. This form of testing is carried out at various stages, starting from the early phases of development right through to the product's launch [46].

Key to usability testing is its ability to provide insight into user interactions with a system, allowing designers to identify what aspects are successful, what users enjoy, what does not work, and what users dislike. By watching users try to complete specific tasks, designers can assess how effective the design or product is and spot any flaws that might have been overlooked.

The process of usability testing encompasses planning by outlining what aspects of the product you wish to test and deciding on the methodology, setting up user tasks by choosing the most critical tasks that align with your objectives, creating realistic scenarios for users to engage with the design, and finally, recruiting testers by identifying your target user group.

A facilitator guides participants through tests, ensuring high-quality data without influencing behaviors, a challenging balance requiring training. Tasks mimic real-life activities, with examples ranging from troubleshooting printer errors to choosing a credit card, crafted to avoid bias through careful phrasing. Participants, ideally actual or similar to real users, are encouraged to think aloud, sharing their thoughts and actions to provide insights into their behaviors, goals, and motivations.

5.2 Testing of Integration

Tested functionality was split into two areas. The first was inside Xperience by Kentico, where the address field is added and set up inside the Forms application. The second was on the website created using Xperience by Kentico, where the added field was rendered. This situation led to the use of people with knowledge of the product and regular internet users in the testing to achieve the best results.

This usability testing aimed to uncover possible design problems with this integration. For this, four testing scenarios were introduced to participants, and they had to complete them. After completion, a Single Ease Question (SEQ) was presented to the people, indicating how challenging the scenario was. The participant graded the difficulty on a scale from 1 to 7, where one was the most complex and seven was the most straightforward [24].

Tested scenarios were:

1. Add Address field to the ContactUs form.
2. For this added Address field, enable address validation, address suggestions, and company name suggestions. Set country restrictions to the Czech Republic and the suggestions' language to Czech.
3. Submit the address "Joštova 144, Brno" into the ContactUs form.
4. Submit the address of Red Hat company into the ContactUs form.

5.3 Evaluation of Testing

In this section, all of the scenarios will be discussed individually, and in the end, a summary of the whole testing will be provided.

Scenario 1: Add Address Field

Overall, this scenario was successful, as all participants completed it.

Two participants required slight hints, as they were unfamiliar with the Forms application and did not know which button adds new fields to the form. This problem is not, however, caused by the design

of integration but rather by the design of the Xperience by Kentico product, which uses two different "Plus" button designs in the Forms application, each serving a different purpose.

Another problem was that few participants were looking for the address field in the Featured category. This field is not present in this category, and it took them time to figure out that another category shows all fields.

Scenario 2: Set-up Address Field

This scenario was the most difficult for participants, with an average score of 5.4.

The first problem was that no category was dedicated to validation and prefiling functionality, so they had to scroll through all the options to find the one they were looking for. This problem was supported by several options being present by default, and this integration added another five.

As a few participants also mentioned after testing, another problem was that the links in explanation texts below two options leading to documentation explaining how to fill in codes of countries and languages were not opening in a new tab. This inability meant they had to force-open the link in a new tab to avoid losing the current options state. This problem is, however, in the product itself, as Xperience by Kentico does not support links that open new tabs in texts due to security reasons.

Scenario 3: Submit address

This scenario was the easiest as all participants completed it quickly with no problems. This fact was also reflected by an average score of 7, the highest possible score. Such a success in this scenario was likely due to users filling in addresses often and being comfortable with using address autocomplete.

Scenario 4: Submit company address

The last scenario was also successful, as all participants submitted the correct value to the form. No one even searched for the company's

address in another tab. Everyone started typing the company's name into the fields and clicked on the correct suggestion.

One problem with this scenario was that there were two suggestions for Red Hat company, as this company has several buildings in Brno. Some participants were thinking about which they should click. This problem was caused by poor scenario design, as the possibility of two similar suggestions was not considered during creation.

Summary

Overall, the usability testing was successful as the average score of all scenarios was 6.25, well above 5.7, which is suggested Kentico's result. All the scenarios and their SEQ scores can be compared in Table 5.1.

Table 5.1: Testing scenarios with their SEQ results

Participant	Scenario 1	Scenario 2	Scenario 3	Scenario 4
1	5	5	7	6
2	6	7	7	7
3	7	5	7	7
4	7	6	7	6
5	6	4	7	6
Average	6.2	5.4	7	6.4

Two main improvements that will be addressed in the future are the addition of a category for validation and prefilling options and the move of this newly created field into the featured category. In addition, if the support for links opening in new tabs is added, this feature will be added to links in explanation texts below options.

6 Conclusion

The goal of this thesis was to investigate options for integrating the address validating and prefilling feature to submission forms in Xperience by Kentico.

Available APIs supporting address validation and prefilling were researched, other competitors' solutions were studied, and feedback was gathered from company partners via a research questionnaire. Based on the research results, Google Maps API was selected for integration into Xperience by Kentico. This API offers good coverage, has many valuable features, and is already well-known to partner companies that use address validation.

Integration of Google Maps API was implemented, offering customers an easy-to-use component for their forms. With many options that can be changed, this component has flexibility and can fit into multiple use case scenarios. Various countries are supported by this feature, allowing users to fill in addresses in different formats while all of the addresses are correctly validated and stored.

Finally, usability testing was performed on this integration. Although testing revealed some usability issues, all participants were able to use this feature effectively, resulting in a successful completion of testing.

A Electronic Attachments

A.1 Demo Website

To try out this integration without any need for setup or configuration, a demo website has been deployed to the Azure platform [47]. The website will be available for only a limited time due to a trial Google Cloud account usage.

Xperience by Kentico administration:
https://xbkgooglemaps.azurewebsites.net/admin/dashboard

- User name: *user@mail.com*
- Password: *yHGlS)HEw@MveHD̂6*

The demo website with the form:
https://xbkgooglemaps.azurewebsites.net/contacts

A.2 Content of Archive

- Integration source code with setup and usage guide
- Tests project
- Example *DancingGoat* project in Xperience by Kentico with referenced integration

A.3 Google Cloud Setup

1. Log into Google Cloud
2. Create a new project
3. Enable the following APIs: Geocoding API, Places API, Maps API, Address Validation API
4. Copy the API key and paste it into *appsettings.json* file later

A.4 Integration Installation

1. Add reference to integration in the root folder of the Xperience by Kentico project:

```
dotnet add reference ..\src\Kentico.
   Xperience.GoogleMaps\Kentico.Xperience.
   GoogleMaps.csproj
```

2. In *Program.cs* file in the local instance, add the following last line:

```
WebApplicationBuilder builder =
   WebApplication.CreateBuilder(args);
builder.Services.AddKentico();
builder.Services.AddXperienceGoogleMaps(
   builder.Configuration);
```

3. In the *appsettings.json* file, add the following lines:

```
"CMSXperienceGoogleMaps": {
  "APIKey": "<Google Maps API key>"
}
```

4. In the *_ViewImports.cshtml* file, add the following lines:

```
@using Kentico.Xperience.GoogleMaps
@addTagHelper *, Kentico.Xperience.
   GoogleMaps
```

5. In the website layout file (for example: *_DancingGoutLayout.cshtml*), add the following line:

```
<google-maps />
```

A.5 Research Questionnaire

At what position do you work?

This question helped determine respondents' general knowledge of our product and programming. Handling submissions from engineering managers, UX designers, testers, or developers requires a different approach, each providing unique perspectives and ideas.

Table A.1: Job positions of the respondents

Responses
Head of Web Development
Developer
Senior Software Engineer
Solution architect
Senior Web Engineer
Principal Full Stack Developer
Web Developer
Architect

Table A.1 shows all the respondents' positions. Most of them are developers, software engineers, or web engineers. They possess vast programming knowledge, showing that they frequently use our product during their website development. Additionally, two architects within this group likely have a strong understanding of website creation on a high-level basis.

Do you use validation rules for fields in your forms?

Answers to this question, as can be seen in Figure A.1, confirmed the hypothesis that all customers need validation rules in their forms to specify better the results they want to obtain from users.

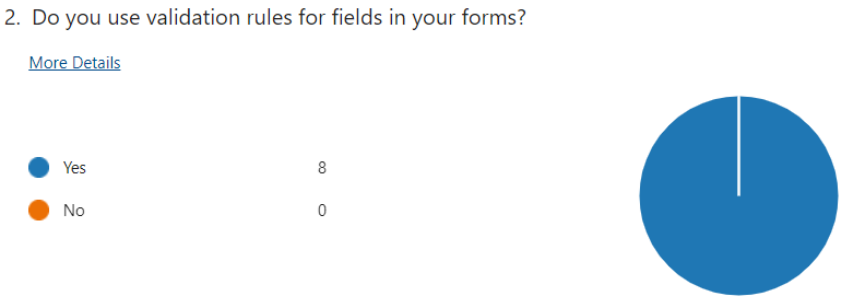

Figure A.1: Usage of validation rules

Have you ever tried creating your own custom validation rule?

Experience creating custom validation rules shows that partners require data sent to them in a specific format they want to control. Meaning that the product's currently available options are not enough for them.

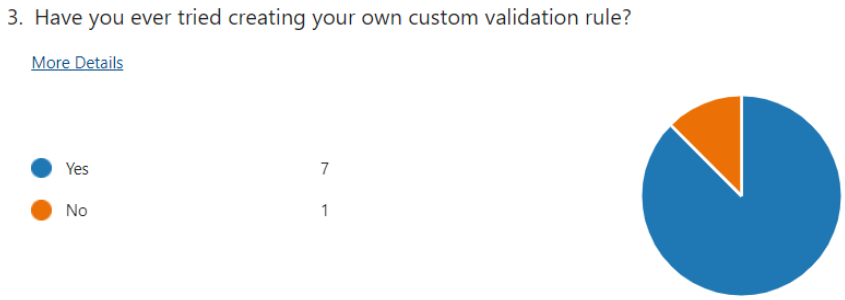

Figure A.2: Creation of custom validation rules

The graph presented in Figure A.2 indicates that many partners desire more control over their data by creating their own custom validation rules. However, this process often requires a developer to gather requirements from content editors and implement the necessary validation rules, which can be time-consuming. To save time and resources, our product can already include support for the most commonly used validation, enabling customers to apply the validation to selected fields using just one person without any programming knowledge.

How do you currently approach validation of addresses in forms?

This question helped determine the importance of address validation for partners and whether they would find it useful.

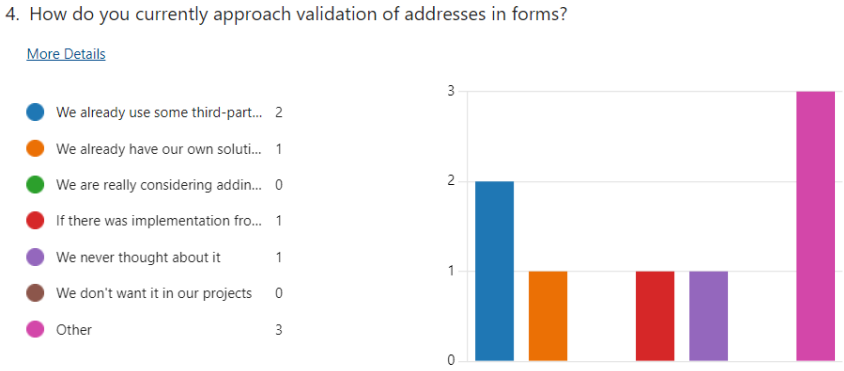

Figure A.3: Current approach to address validation

As the graph in Figure A.3 shows, two partners already have some third-party solutions for address validation. One has his custom solution, and another would use the solution created in this project if it was in Xperience by Kentico. One respondent never thought about the validation of addresses. The other three answers mentioned this: Usage of Google Maps API and Azure Reverse Geocoding, usage of many different APIs for specific scenarios, and the fact that this is not something they usually need. This shows that most of the partners need address validation in their projects.

When filling some online form with address validation, have you ever encountered any problems with it?

Learning from mistakes others did and avoiding them is very important, so the following two questions helped with identifying critical problems that need to be addressed during the design phase. Only respondents who answered "Yes" to the first question were shown the second question, resulting in only seven responses to the second question.

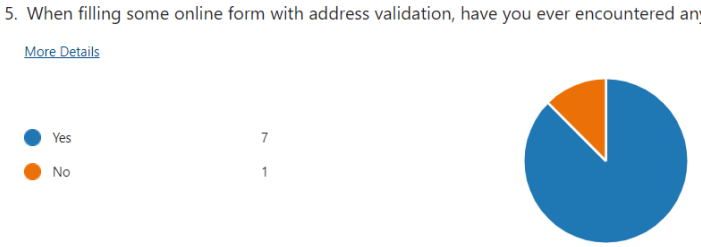

Figure A.4: Encountering problems with address validation

Figure A.4 highlights many problems with validated fields for addresses, and many people have to deal with them.

What problems did you encounter?

Table A.2: Problems with validation

Responses
New addresses may not exist and need manual entry.
Address matches are not always alphabetically ordered.
Address recognition issues with foreign addresses.
Post-back interferes with typing addresses.
Different address formats for different countries.
New streets are not recognized, causing validation issues.
Problems with postal codes and states.
API is not responding.

From looking at Table A.2, it is apparent that there are many different problems. Those answers are valuable insights that will be helpful during the design phase and implementation.

How would you like to work with possibly invalid addresses submitted into your form?

This seemed to be the most problematic and hard-to-decide problem during the first discussions and thoughts about this project and its implementation.

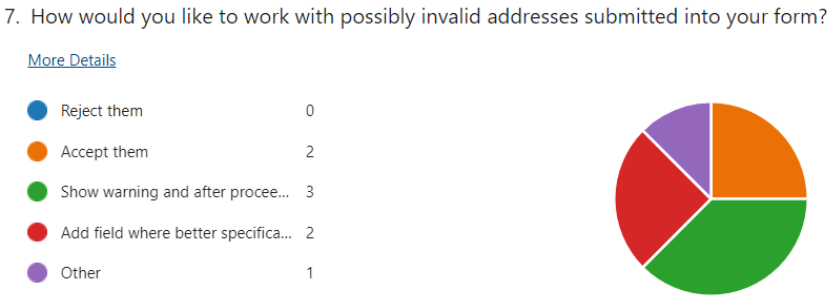

Figure A.5: Handling of invalid addresses

As shown in Figure A.5, the respondents had varying opinions regarding the problem. The other response mentions that addresses are typically looked up using third-party API and only valid are accepted. None of them favored outright rejection of invalid addresses. Instead, they preferred accepting such addresses and possibly indicating that the address is not in the API used.

Are you collecting the names of your customers' companies through forms?

The following three questions helped with determining the importance of collecting company names. The following two questions appeared to respondents only if they answered "Yes" in this one.

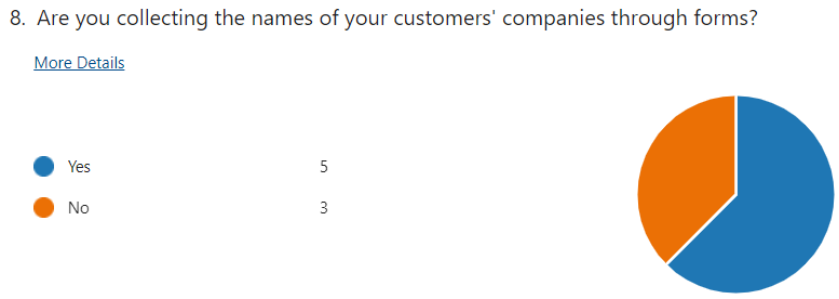

Figure A.6: Collecting customers' names

As Figure A.6 indicates, collecting company names is not as common as collecting customer addresses. The reason for this is that addresses are helpful to collect from companies and individual customers, whereas collecting the name of a company from a regular customer does not make sense.

How often do you collect those company names?

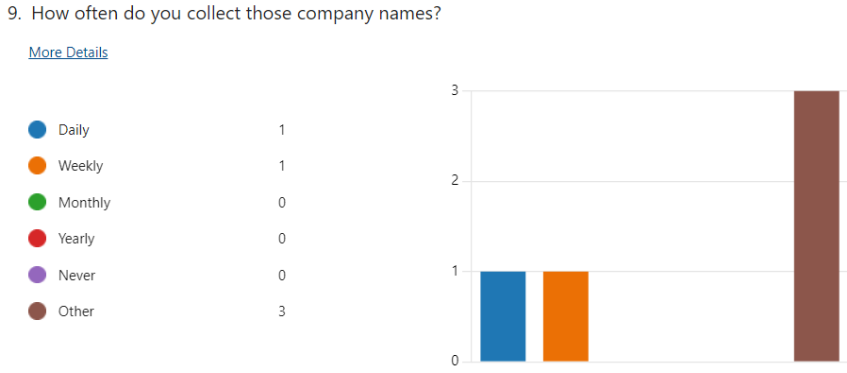

Figure A.7: Frequency of company names collecting

As per the Other section in Figure A.7, all responses mention that the frequency of collecting answers depends on the project. However, in general, they are not collected very often. This illustrates that only a few partners collect names frequently, while most of them collect names rarely or not at all.

How do you work with these collected company names?

This question addressed the possible need to validate if the company entered into the form exists.

Figure A.8: Handling of company names

In Figure A.8, only one answer, hidden behind the Other category, mentions that respondents validate company names. However, this response only refers to a single project where such validation was necessary using a third-party database. This suggests that most customers do not consider company name validation a priority, as they rarely need it.

How do you currently approach getting address suggestions based on entered company name in your forms?

When this project was created, one of the mentioned features was those suggestions. This question aims to determine if the company partners already have a solution and whether they find this problem essential.

Figure A.9: Handling of addresses based on company names

The results shown in Figure A.9 indicate that the respondents have not yet implemented a similar functionality in their products. Other answers mention the possible usage of Google Maps API and Loqate for company name lookup. However, they have expressed interest in having this feature included in Xperience by Kentico.

If you have anything else that you would like to add, please feel free to do so.

Table A.3: Additional feedback

Notes
Regular updates of address data are important.
Addresses change over time.
Google autocomplete can be easily used.
This feature requires paid validation services.
Usefulness depends on the project.
Xperience by Kentico extension would be beneficial.
Do not reinvent the wheel.
Existing solutions are available.
Internationalization is essential.
Locale-specific formats are beneficial.

All the suggestions displayed in Table A.3 will be beneficial during the design phase. Additionally, there were valuable recommendations for APIs to use, which will be explored further during the research phase on APIs for this project.

At the end of the questionnaire, there was also an option to provide an email address so that a beta version of the project could be sent to the respondent. Six respondents left their email addresses, indicating that a majority of the participants are interested in testing out this new feature before it is officially released.

Bibliography

1. WESTERVELD, Dave. *API Testing and Development with Postman*. Packt Publishing, 2021. ISBN 9781800569201.
2. MASSE, Mark. *REST API Design Rulebook*. O'Reilly Media, Inc., 2011. ISBN 9781449310509.
3. *What is API Client?* [online]. Postman [visited on 2024-05-01]. Available from: https://www.postman.com/api-platform/api-client/.
4. *What is an API call?* [online]. Cloudflare [visited on 2024-05-01]. Available from: https://www.cloudflare.com/learning/security/api/what-is-api-call/.
5. FIELDING, R.; GETTYS, J.; MOGUL, J.; FRYSTYK, H.; MASINTER, L.; LEACH, P.; BERNERS-LEE, T. *RFC2616: Hypertext Transfer Protocol – HTTP/1.1*. USA: RFC Editor, 1999.
6. *Work with API response data and cookies in Postman* [online]. Postman [visited on 2024-05-01]. Available from: https://learning.postman.com/docs/sending-requests/response-data/response-data/.
7. BIEHL, Matthias. *API Architecture*. API-University Press, 2015. ISBN 9781508676645.
8. AL-KHAMISI, Zakariya. *Top 6 API Architecture Styles for Modern Software Development* [online]. Medium, 2023-10-06 [visited on 2024-05-10]. Available from: https://zikazaki.medium.com/top-6-api-architecture-styles-for-modern-software-development-d35752c7f9aa.
9. *What is Address Validation?* [online]. Amsterdam, Netherlands: TomTom, 2020-06-05 [visited on 2024-04-21]. Available from: https://www.tomtom.com/newsroom/explainers-and-insights/what-is-address-validation/.
10. GUERMAZI, Yassine; SELLAMI, Sana; BOUCELMA, Omar. A RoBERTa Based Approach for Address Validation. 2022. Available also from: https://link.springer.com/chapter/10.1007/978-3-031-15743-1_15.

11. *Address Cleansing* [online]. Experian Data Quality [visited on 2024-04-21]. Available from: https://www.edq.com/address-cleaning/.

12. *What is Address Verification?* [online]. Mapbox [visited on 2024-04-21]. Available from: https://www.mapbox.com/insights/address-verification.

13. ROBERTSON, Ronald E.; JIANG, Shan; LAZER, David; WILSON, Christo. Auditing Autocomplete: Suggestion Networks and Recursive Algorithm Interrogation. *Proceedings of the 10th ACM Conference on Web Science*. 2019. Available also from: https://doi.org/10.1145/3292522.3326047.

14. FLEISHER, Craig; BENSOUSSAN, Babette. *STRATEGIC AND COMPETITIVE ANALYSIS: Methods and Techniques for Analyzing Business Competition*. Prentice Hall, 2002. ISBN 9780131918720. Available also from: https://www.researchgate.net/profile/Craig-Fleisher/publication/265224384_Strategic_and_Competitive_Analysis_Methods_and_Techniques_for_Analyzing_Business_Competition/links/54dccbb10cf25b09b912ce82/Strategic-and-Competitive-Analysis-Methods-and-Techniques-for-Analyzing-Business-Competition.pdf.

15. JONASSON, H. *Determining Project Requirements*. 1st. Auerbach Publications, 2007. ISBN 9780429119293. Available also from: https://doi.org/10.1201/9781420045031.

16. *Place Autocomplete* [online]. Mountain View, CA, USA: Google for Developers [visited on 2024-05-01]. Available from: https://developers.google.com/maps/documentation/javascript/place-autocomplete.

17. *Send an address validation request* [online]. Mountain View, CA, USA: Google for Developers [visited on 2024-05-01]. Available from: https://developers.google.com/maps/documentation/address-validation/requests-validate-address.

18. *Understand a basic address validation response* [online]. Mountain View, CA, USA: Google for Developers [visited on 2024-05-01]. Available from: https://developers.google.com/maps/documentation/address-validation/understand-response.

19. *About | Kentico* [online]. Brno, Czech Republic: Kentico [visited on 2024-04-07]. Available from: https://www.kentico.com/about.

20. *What is a Content Management System (CMS)?* [online]. Los Angeles, CA, USA: Kinsta, 2023-10-18 [visited on 2024-04-15]. Available from: https://kinsta.com/knowledgebase/content-management-system.

21. *Types of CMS: A Comprehensive Guide* [online]. New York, NY, USA: DesignRush [visited on 2024-04-15]. Available from: https://www.designrush.com/agency/web-development-companies/trends/types-of-cms.

22. *What is a Digital Experience Platform?* [online]. Los Angeles, CA, USA: Liferay [visited on 2024-04-15]. Available from: https://www.liferay.com/resources/l/digital-experience-platform.

23. *GraphQL | A query language for your API* [online]. Global: GraphQL, 2024-03-30 [visited on 2024-04-07]. Available from: https://graphql.org/.

24. DAVIS, H. *Search Engine Optimization*. O'Reilly Media, 2006. ISBN 9781491905883.

25. *What is SaaS? Software as a Service | Microsoft Azure* [online]. Redmond, WA, USA: Microsoft Azure, 2024-03-30 [visited on 2024-04-07]. Available from: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-saas.

26. *Digital experience platform | Kentico Xperience by Kentico* [online]. Brno, Czech Republic: Kentico [visited on 2024-04-07]. Available from: https://www.kentico.com/platforms/xperience-by-kentico.

27. *ASP .NET Core* [online]. Redmond, WA, USA: .NET [visited on 2024-04-07]. Available from: https://dotnet.microsoft.com/en-us/apps/aspnet.

28. *MVC - MDN Web Docs Glossary: Definitions of Web-related terms* [online]. Mountain View, CA, USA: MDN Web Docs [visited on 2024-04-07]. Available from: https://developer.mozilla.org/en-US/docs/Glossary/MVC.

29. NUNNALLY, Brad; FARKAS, David. *UX Research: Practical Techniques for Designing Better Products*. O'Reilly Media, Inc., 2016. ISBN 9781491951293. Available also from: https://books.google.com/books/about/UX_Research.html?id=sfOeDQAAQBAJ.

30. MARTINELLI, Suellen; LOPES, Larissa Albano; ZAINA, Luciana A. M. UX research in the software industry: an investigation of long-term UX practices. 2022. Available also from: https://typeset.io/papers/ux-research-in-the-software-industry-an-investigation-of-37z1b4q0.

31. *Problem Statements in UX Discovery* [online]. Fremont, CA, USA: Nielsen Norman Group, 2021-08-22 [visited on 2024-03-10]. Available from: https://www.nngroup.com/articles/problem-statements/.

32. KAASINEN, Eija; ROTO, Virpi; HAKULINEN, Jaakko; HEIMONEN, Tomi; JOKINEN, Jussi P. P.; KARVONEN, Hannu; KESKINEN, Tuuli; KOSKINEN, Hanna; LU, Yichen; SAARILUOMA, Pertti; TOKKONEN, Helena; TURUNEN, Markku. Defining user experience goals to guide the design of industrial systems. *Behaviour & Information Technology*. 2015, vol. 34, no. 10, pp. 976–991. Available from DOI: 10.1080/0144929X.2015.1035335.

33. *User Research Questions - User Interviews* [online]. Cambridge, MA, USA: User Interviews [visited on 2024-03-10]. Available from: https://www.userinterviews.com/ux-research-field-guide-chapter/user-research-questions.

34. *A Guide to Using User-Experience Research Methods* [online]. Fremont, CA, USA: Nielsen Norman Group, 2022-08-21 [visited on 2024-03-10]. Available from: https://www.nngroup.com/articles/guide-ux-research-methods/.

35. PATTEN, Mildred L. *Understanding Research Methods: An Overview of the Essentials*. 9th. Routledge, 2013. ISBN 9781315266312. Available also from: https://doi.org/10.4324/9781315266312.

36. ROOPA, S; RANI, MS. Questionnaire Designing for a Survey. *Journal of Indian Orthodontic Society*. 2012, vol. 46, no. 4_suppl1, pp. 273–277. Available from DOI: 10.5005/jp-journals-10021-1104.

37. *Form design: multiple inputs versus one input* [online]. Adam Silver, 2020-05-18 [visited on 2024-04-01]. Available from: https://adamsilver.io/blog/form-design-multiple-inputs-versus-one-input/.

38. *Google Maps Platform Coverage Details* [online]. Mountain View, CA, USA: Google for Developers [visited on 2024-04-01]. Available from: https://developers.google.com/maps/coverage.

39. PACHECO, Juan Fernando. *4D UX methodology - Juan Pacheco | Juan Fernando Pacheco Design Thinking the 4D UX methodology* [online]. Medium, 2018-02-19 [visited on 2024-04-21]. Available from: https://medium.com/juanpacheco/design-thinking-the-4d-ux-methodology-e621adc3339c.

40. SILVESTRI, Gabriel. *UX Design Steps Infographic Illustration* [online]. Dribbble [visited on 2024-04-27]. Available from: https://dribbble.com/shots/4512879-UX-Design-Steps-Infographic-Illustration.

41. *Design Patterns: Factories* [online]. Redmond, Washington, United States: Microsoft, 2017-08-10 [visited on 2024-04-27]. Available from: https://learn.microsoft.com/en-us/shows/visual-studio-toolbox/design-patterns-factories.

42. *.NET Dependency Injection - .NET Documentation* [online]. Redmond, Washington, United States: Microsoft, 2024-03-15 [visited on 2024-04-27]. Available from: https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection.

43. *Form Components* [online]. Brno, Czech Republic: Kentico [visited on 2024-04-27]. Available from: https://docs.kentico.com/developers-and-admins/development/builders/form-builder/form-components.

44. *Tag Helpers in ASP.NET Core* [online]. Redmond, Washington, United States: Microsoft, 2023-07-25 [visited on 2024-05-08]. Available from: https://learn.microsoft.com/en-us/aspnet/core/mvc/views/tag-helpers/intro?view=aspnetcore-8.0.

45. *Software Testing: Mock Testing - David Christianto* [online]. Medium, 2023-10-11 [visited on 2024-04-27]. Available from: https://medium.com/@david.christianto05/software-testing-mock-testing-76f572b58936.

46. RIIHIAHO, S. Usability Testing. In: *The Wiley Handbook of Human Computer Interaction*. Ed. by NORMAN, K.L.; KIRAKOWSKI, J. 2018. Available from DOI: 10.1002/9781118976005.ch14.

47. *Cloud Computing Services | Microsoft Azure* [online]. Redmond, Washington, United States: Microsoft [visited on 2024-05-10]. Available from: https://azure.microsoft.com/cs-cz.