TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Mathias Kivi 193815IADB

MERCH STORE

Project for Building Distributed Systems (ICD0009) course

Supervisor: Andres Käver

Tallinn 2021

Author’s declaration of originality

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

Author: Mathias Kivi

[dd.mm.yyyy]

Table of contents

1 Introduction

The goal of this project is to create a web store application. In recent years I have seen more and more people getting into an influencer career and once they have gained some popularity, they want to offer their fans products with logos on them, also known as merchandise.

Now more than ever is online shopping blooming, and since influencers' following can grow into huge numbers, it's simply not reasonable to sell merchandise outside of e-commerce. However, the biggest issue with opening an online store is to find a platform that would be cheap to maintain and easy to set up, which means that they mostly have two options: using a monthly subscription-based platform such as Shopify or making an order for a custom-made online shopping platform from a software developing company. Both of these options are pricey, which forces you to sell the products at a higher price, which leads to even fewer amount of orders – this is especially true if you have a smaller following.

This project aims to provide support for the most basic features that an online shopping platform should have, including but not limited to product list page, product page, shopping cart, order placement, order tracking and multilingual support.

The product list pages are categorized and can be sorted by popularity, title (ascending and descending) or by price (high-to-low and low-to-high). The shop managers can add new products, add pictures about that product and manage stock availability. Users can place new orders and track their statuses, like whether the payment has been processed or has the order shipped yet. The items will be shipped by using parcel machines only – there will not be options for courier or in-store pickup.

The objective of this project is to create an online store application as a minimal viable product, aimed at selling specific items, such as merchandise.

2 Analysis

2.1 Application overview

The plan is to create an online store web application, using both ASP.NET Core MVC framework, to create a conventional static web page, and Vue.js JavaScript framework, to create a single page application, which retrieves and sends data through API endpoints. API endpoints are created with ASP.NET Core framework.

The users can browse products, which includes viewing product details such as description and pictures, sorting them by category and/or best sellers, product's name and price.

The sellers can add new products with details such as name, description and pictures. Sellers can also view orders and edit their details, like status and tracking details.

The initial plan was to use Maksekeskus as the payment provider, but it turned out that they offer services only for companies or persons who are self-employed. After some research I decided to use PayPal as the payment method, because they provide a well thought out and easy to use checkout functionality for private persons as well.

2.2 Scope of work

It's clear that creating a fully featured online store platform would be too much work to fit in this course, so I have made some bullet points about what will be created and what could be implemented later on.

Features that will be created:

- Sellers can add products with details like description and pictures.
- Users can browse products and sort them by category and/or best sellers, product's name and price.
- Users can add products to their shopping cart without logging in.
- Users can specify parcel machine provider and location.
- Users can checkout with PayPal Checkout service.
- Sellers can add tracking identifier/link to orders.
- Users can track orders by tracking identifier/link, specified by sellers.

Features that could be added later on:

- Sellers can add product-specific color and size options.
- Sellers can make per-product and/or shopping cart discounts.
- Users can add products to their wishlist.
- Users can add reviews to products that they have bought.

2.3 Application structure

In order to create an application which can be easily managed and improved years later, it's important to emphasize the code structure. Highly interdependent code is difficult to modify and fragile to changes. One of the ways to make it more independant is to use a repository pattern, which I tried to follow while creating this application.

The repository pattern is a strategy for making data access abstract, which gives a number of benefits.[1] The most remarkable of them being: a substitution point for unit tests and a flexible architecture. The latter means that if you want to modify the data access logic, you do not need to change the higher levels as well.

3 Diagrams

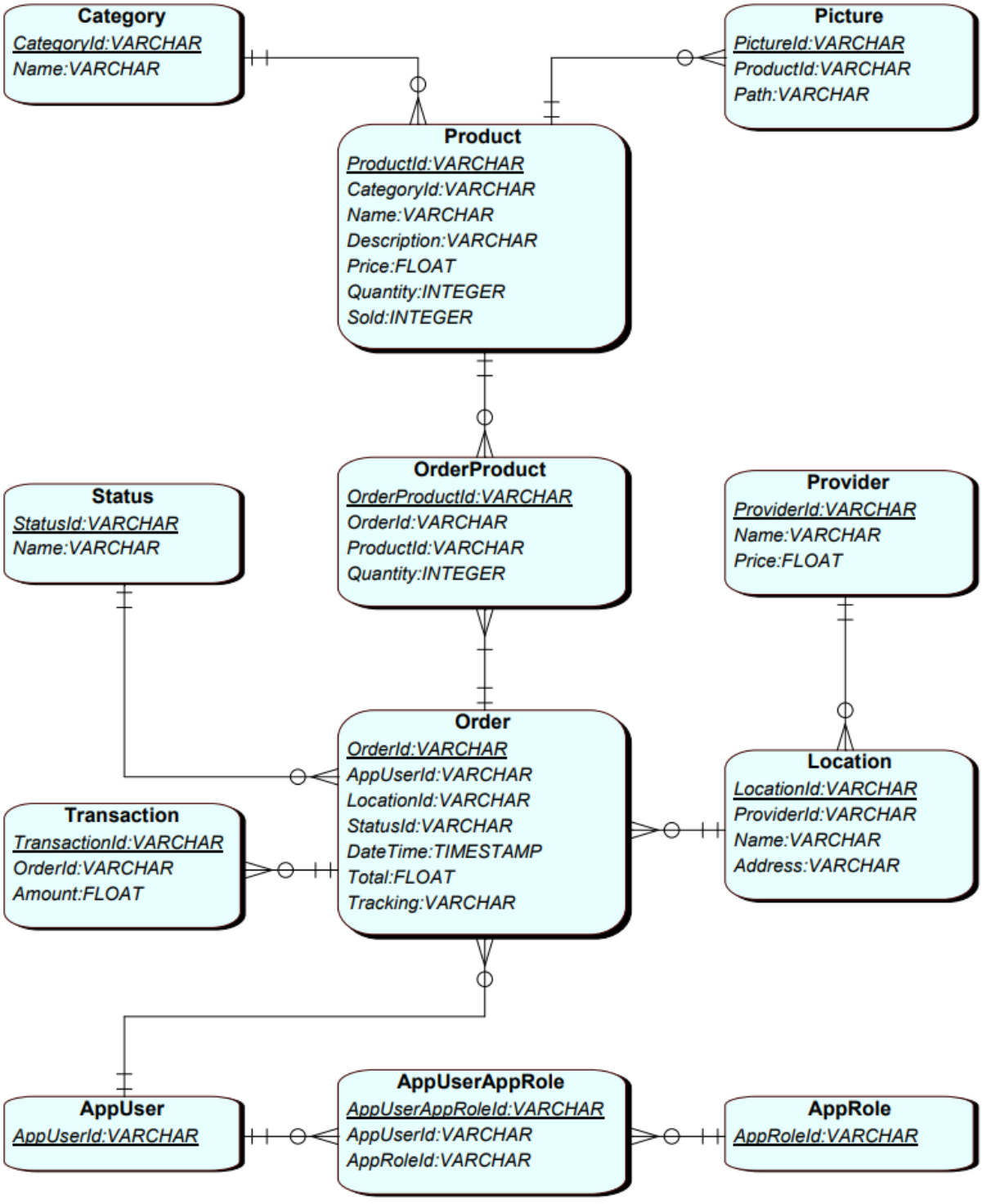

Figure 1. Entity relationship diagram.

4 Summary

The purpose was to create an online store application. I got this idea, because at the time I had recently seen an ugly online store and wanted to try if I could make an online store myself, preferably a nicer one.

When I had sorted out the project's goal, it was time to learn more about how I could implement the payment solution into my website. After some research, I found out that all of the providers that supported Estonian banks required you to have a company, so the best alternative seemed to be PayPal. It is widely used by Estonians and it has a well thought out and easy to use SDK for implementing their checkout functionality to your website.

Programming and creating the back- and front-end worked out fine and there were no issues. In both cases, optimality had to be taken into account. Poorly optimized code and cluttered user interface can affect application's performance and usability in a negative manner.

In the future, the application could be improved by implementing features listed earlier. If were are a smaller creator, then this project could be a viable option. However, if you have gained a bigger following, they probably expect a little bit more.

Used materials

1. J. Petrakovich. *The WHY Series: Why should you use the repository pattern?*. https://makingloops.com/why-should-you-use-the-repository-pattern/ (27.04.2021)
2. *Design the infrastructure persistence layer*. https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design (27.04.2021)