Tartu Ülikool

Sotsiaalteaduste valdkond

Narva kolledž

Infotehnoloogiliste süsteemide arenduse õppekava

Vladimir Galitski

Autopesula väikeettevõtte töölaua rakenduse arendamine koos kassa- ja juhtimiskonsooliga

Bakalaureusetöö

Juhendaja: Andre Sääsk, M. Sc.

Narva 2025

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Vladimir Galitski

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose
   „Autopesula väikeettevõtte töölaua rakenduse arendamine koos kassa- ja juhtimiskonsooliga“
   mille juhendaja on Andre Sääsk ja Chen-Wan Lin

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, alates ***06.06.2025*** kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Vladimir Galitski
06.06.2025

CONTENTS

DICTIONARY

IS – Information System

WMS – Warehouse Management System

OMS – Order Management System

ERP – Enterprise Resource Planning

CRM – Customer Relationship Management

SCM – Supply Chain Management

HRM – Human Resource Management

HRMS – Human Resource Management System

MRP – Manufacturing Resource Planning

INTRODUCTION

The field of car wash management encompasses a wide variety of business processes and specifications, making it nearly impossible to find a single off-the-shelf application that fully satisfies the unique requirements of a small car wash business. Consequently, the solution under development will account for the specific workflows of a car wash and adapt to its individual needs, ensuring both flexibility and scalability.

Problem

There are no good applications on the market that meet all specific requirements. This work will be dedicated to developing an IT solution to automate a car wash's interactions within a company.

Goal

The primary objective of this work is to develop a desktop application for a small car wash company that integrates both a cash register and a management console. To ensure a comprehensive understanding of the project's requirements and tasks, this work will begin with a brief analysis of the historical evolution of information systems. This analysis will examine key contributions made by various scholars and practitioners, outlining who played vital roles and when significant achievements in the development of this concept occurred.

The thesis will provide an analysis of the evolution of major business information systems, such as OMS (Order Management Systems), CRM (Customer Relationship Management), ERP (Enterprise Resource Planning), MRP (Material Requirements Planning), among others. Definitions and functional descriptions of these systems will be presented to establish a foundational context for their application in modern business environments.

Tasks

The objectives of this work include:

- Investigating existing applications available on the market.
- Justifying the necessity for developing a new application.

- Selecting appropriate tools for application development.
- Creating a comprehensive list of required functionalities.
- Developing the program responsible for managing cash register operations.
- Designing the necessary user interface windows and implementing their functionalities.
- Incorporating transportation data as required.

Structure

The structure of this thesis will be organized into three main chapters. The first chapter will provide an overview of the historical development of various information systems, including the definitions and classifications that have been proposed by different scholars and practitioners over time. The second chapter will focus on analyzing the functional and technical requirements of the application. It will also include a competitive analysis of existing car wash businesses in Narva, as well as a review of currently available software solutions on the market. The third chapter will present a description of the development process, outlining the tools that will be used, the key implementation steps, and the reasoning behind design decisions.

1. HISTORY

1.1. History of Information Systems

The study of Information Systems (IS) has been going on for many years. There are many scholars and practitioners who have contributed to the development of the idea. This chapter provides an overview of the evolution of the concept of IS, highlighting key contributions and definitions that have influenced its modern interpretation.

1.1.1. Early Conceptualizations

In the early periods of the development of organizations, the process of defining an information system was a slow process and was not associated with any one author. The concept originated from the management and computer science and operations research disciplines as organizations started to use technological support in their decision-making processes. However, the first systematic academic attempt to define what an information system is, was made in 1958 in the article “Management in the 1980s” by Harold Leavitt and Thomas Whisler. They defined information systems as the application of technologies and procedures in order to produce and distribute information to support management decision making. (Harold Leavitt, 1958, p 41)

1.1.2. Contributions from the 1970s

The concepts of information systems were further developed in the 1970s and 1980s through several important works:

Gordon B. Davis (1974)

Davis is one of the earliest scholars in the discipline, and he offers a systematic definition of an information system as a system that collects, processes, stores and distributes information to support management decision making (Gordon B. Davis, 1985, p. 6). His contribution especially established the fundamental framework that was to guide other researchers and practitioners in their work.

Richard Nolan & Cyrus F. Gibson (1974)

In his work “Managing the Four Stages of EDP Growth” Richard Nolan outlined the four stages of the information system life cycle and recommended solutions for problems that might occur at each stage. He termed these stages as Initiation, Expansion,

Formalization and Maturity. Nolan's model shows how organizations start with limited attempts at using electronic data processing to the point where information systems are fully incorporated into the strategic management process.

1.1.3. Contributions from the 1980s

James Nunamaker (1983)

IS were seen to play a crucial role as a tool in the decision-making process by Nunamaker. In his work, he identified the alliance between the technological infrastructure and the needs of the management to emphasize the role of IS in the improvement of decision making. His contributions also extended the concept of information systems to areas that were previously associated with data processing. A DSS for Systems Analysis and Design (c. W. Holsapple and A. B. Whinston, 1983, p251.)

Ralph Stair & George Reynolds(1993)

In their book "Principles of Information Systems", they defined an information system as a system that acquires, transforms, stores and distributes information in order to support management decision making and business operations.(Ralph Stair & George Reynolds, 2008, p 4)

1.1.4. Further Evolution of Business Information Systems

At the same time, academic contributions were made, the business world also saw the rise of various specialized information systems for particular operational purposes. Some of the systems that emerged include Order Management Systems (OMS), Customer Relationship Management (CRM), Enterprise Resource Planning (ERP) and Material Requirements Planning (MRP) and each of these systems was defined based on its particular purpose:

1.2. The Evolution of CRM: A Historical Perspective

Customer Relationship Management (CRM) has been developing for several decades from a marketing concept to a modern business leadership tool. This change has been encouraged by the key ideas of relationship marketing, the shift to one-to-one marketing approaches, and the development of technology that enabled companies to deal with customer data. In this chapter are described the contributions of Leonard Berry, Don Peppers and Martha Rogers, and Tom Siebel in developing and popularizing CRM.

1.2.1. First steps: Leonard Berry and Relationship Marketing

It was at the beginning of the 1980s that the shift of focus in marketing from individual transactions to the development of customer relationships took place. Leonard Berry was one of the earliest researchers in this field, claiming that the competitive advantage of service organizations depends on customer retention and satisfaction. Berry's work, especially his chapter "Relationship Marketing" in the book Emerging Perspectives on Services Marketing (1983), suggested that companies should invest in identifying and nurturing long-term relationships with customers instead of looking for new customers (Leonard Berry, 1983, p 25). This theoretical framework formed the basis for the development of the CRM strategies that are used today.

Berry's study focused on the quality of service, the length of the service, and what the client wanted, and it formed the basis for the later developments in customer relationship management. (Leonard Berry, 1983, p. 26-27) The focus on the relationship with the customer in the long run has since been adopted by a large number of marketing strategies, which was able to change the way that organizations interact with their customers.

1.2.2. One-to-One Marketing: Don Peppers and Martha Rogers

Building upon the ideas of relationship marketing, Don Peppers and Martha Rogers introduced the concept of one-to-one marketing in the early 1990s. Their influential book, "The One-to-One Future" (1992), argued that businesses should treat each customer as an individual rather than a member of a mass market segment. This idea was firstly introduced in book "The One-to-One Future", and in their book "Managing Customer Relationships", which develops further their ideas. (Peppers & Rogers, 2004, p. 6)

Peppers and Rogers also gave general definition of CRM - "as a set of business practices designed, simply, to put an enterprise into closer and closer touch with its customers, in order to learn more about each one". (Peppers & Rogers, 2004, p. 6)

1.2.3. The Technological Revolution: Tom Siebel and Siebel Systems

When ideas started to be applied in business, technology played a key role in the evolution of relationship marketing into actual CRM systems. In the 1990s, Tom Siebel, founder of Siebel Systems, also played an important role in the development of

computer database technologies. Siebel Systems first appeared in 1993 (John Deighton, 2003, p. 1) and was a version of the current CRM software solutions that enable organizations to gather, store and analyze customer data to an extent that had not been possible before.

A good example of a case study prepared by Harvard Business School in 2003 is "Siebel Systems: Anatomy of Sales" case study which describes how his company changed the approach to managing customers. This article explained how with the help of Siebel's software tools, companies could manage data from many sources, perform routine customer activities, and develop one-on-one relationships with their customers over time.

1.2.4. Modern Definition of CRM

The concept of Customer Relationship Management (CRM) has further developed in the last few years to include the current technological advancements, data analysis, and business strategies. Different current studies have offered new definitions of CRM that focus on its role in business vision, customer interactivity, and competitive leadership.

In the article "Customer Relationship Management: A Literature Review Approach" (2023) the authors review various CRM definitions and stress on the strategic aspects. They define CRM as "a conceptual approach to the management of customer interactions with the goal of enhancing customer satisfaction, loyalty, and profit through the use of technology and data analysis" (Al-Homery et al., 2023, p. 30). This view of CRM emphasizes the technology-based tool and also the business concept that links marketing, sales, and customer service to develop the bond with the customer.

A more general view of CRM is offered in the 2017 paper "Customer Relationship Management (CRM): Philosophy and Its Significance to the Enterprise". The authors explain that CRM is not just software applications, but a business philosophy that incorporates marketing strategies for handling transactions and customer data, customer-centric strategies for customer service and support, and data analysis to enhance customer interactions (Idzikowski et al.,2019, p. 4). This interpretation also supports the idea that CRM is not only a technological system but a basic concept of the way that businesses should operate by focusing on the customer and their value for the long term.

A similar view is offered in the article "CRM-система как инновационный инструмент повышения конкурентоспособности организации", published in the Scientific

Journal of ITMO University. The authors of the article give the following definition of CRM systems: “an instrument … with which the company’s communication with the client is controlled and forecasted and tracked,” (Klochkova & Bebyakina, 2019, p. 178). In this study, CRM is discussed in the context of digital transformation and organizational innovation as well.

1.3. History of ERP Development

The history of Enterprise Resource Planning (ERP) systems can be linked to the history of production planning as far back as 1960. A very important milestone in this regard was the creation of Material Requirements Planning (MRP) whose prerequisites can be attributed to system dynamics studies. It is possible that without the appearance of MRP, the integrated approach of the modern ERP systems could have never been developed.

1.3.1. Prerequisites to ERP: Jay W. Forrester’s Contributions

In his vital work Industrial Dynamics (1961), Jay W. Forrester introduced key concepts of industrial dynamics and feedback loops in industrial processes. He was the first to analyze how the manufacturing systems could be understood and managed with the help of computer-based models (Forrester, 1965, p. 13). Forrester’s work identified the need for systematic and automated planning in production (Forrester, 1965, p. 13) thus providing the conceptual foundation for the development of ERP systems.

1.3.2. The Genesis of the ERP Concept

The history of ERP systems can be linked to the advancement of production planning systems. The first explicit definition of what was to be called an ERP system was given by Lee Wylie in his research report ERP: A Vision of the Next Generation MRP II. Research Report. In this important work, Lee Wylie outlined the idea of a highly developed system that went beyond the conventional MRP by incorporating other business areas to support more holistic planning and control and claimed that ERP is the natural progression from the concepts identified by MRP.

This concept was rather early to enable the integrated approach that is characteristic of current ERP systems. Olhager in his work “Evolution of Operations Planning and Control: From Production to Supply Chains” (Lund University, Department of Industrial Management and Logistics, Sweden) while outlining the history of the development of

ERP systems also cited the work of Lee Wylie and stated that he was the first to define the term ERP system. In that publication, he pointed out that without the development of MRP systems, there could be no ERP systems. (Olhager, 2013, p. 6836-6843.)

1.3.3. Modern Definitions of ERP

As the ERP systems progressed, the subsequent studies have revised their meanings to suit the technological and strategic business management aspects.

Klaus, Rosemann, and Gable (2000)

Klaus, Rosemann and Gable (2000) in the article What is ERP? Developed one of the most influential modern definitions of ERP in an article, published in Information Systems Frontiers. They define ERP as an effective and integrated software solution that is used to manage and execute business processes. (Helmut et al., 2000, p. 3) The authors of this article state that ERP systems are the informational infrastructure of an organization that provides for the real time data transmission and improvement of the decision-making processes throughout the enterprise.

Vinay Chauhan and Jasvinder Singh (2017)

Expanding the idea further, Chauhan, and Singh (2017) in their study Investment in "Enterprise Resource Planning Systems for Service Performance in Tourism and Hospitality Industry" have viewed ERP systems as strategic decision support tools. Their research shows that the implementation of ERP systems has the potential of enhancing productivity and overall business performance. Moreover, they gave a detailed definition of ERP as "business strategies and enabling software that integrate manufacturing, financial and distribution functions to dynamically balance and optimize enterprise resources.". (Chauhan & Singh, 2017, p. 58)

1.4. History of HRM and HRMS

This chapter explores the evolution of Human Resource Management (HRM) systems, tracing their development from early conceptualizations in the 1980s to the emergence of strategic HRM in the 1990s. By examining seminal works and influential scholars, this chapter illustrates how the field evolved from an operational focus to a strategic cornerstone of organizational success.

1.4.1. Early Developments in HRM (1980s)

Around the same time, another revolutionary perspective emerged with the Oxford Model of HRM. This model was articulated in the seminal work Managing Human Assets by Michael Beer, Bert Spector, Paul R. Lawrence, D. Quinn Mills, and Richard E. Walton. The Oxford Model provided a definition for HRM. They wrote that HRM is the system which is driven by 4 policies: Employee influence (management of employee), human resource flow (combination of number of peoples and their skills for a task), reward systems and work systems (workflow management) (Beer et al., 1984, p. 7-10). They also gave a definition – HRM is a system which involves all management decision and manage relationships between organization and employees (Beer et al., 1984, p. 1)

1.4.2. HRM and HRMS modern Definition

According to Michael Armstrong in A Handbook of Human Resource Management Practice (2006), modern human resource management is defined as follows:

"Human resource management is defined as a strategic and coherent approach to the management of an organization's most valued assets – the people working there who individually and collectively contribute to the achievement of its objectives." (Armstrong, 2006, p. 3). And this means that HRMS system is a system which manages people

This definition reinforces the view that HRM is not merely an administrative function, but rather a strategic and integrated system. Armstrong's perspective emphasizes the importance of managing human capital as a key asset, aligning HR strategies with overall business objectives, and ensuring that every individual within the organization contributes to its success.

1.5. The History of Supply Chain Management

Supply Chain Management (SCM) has evolved significantly over the past few decades – from a narrow focus on logistics to a comprehensive, strategic discipline that integrates various business functions. This chapter reviews the seminal works that have shaped SCM up to the mid-2000s, outlining the key contributions of leading scholars and practitioners.

1.5.1. Pioneering Work (1982)

In 1982, Oliver, R. Keith, and Michael D. Webber published the influential article "Supply-Chain Management: Logistics Catches Up with Strategy" in Outlook. Widely recognized as one of the first attempts to define the concept of SCM.

Douglas M. Lambert later cited this pioneering work as a critical reference point, saying that it was the first work where SCM definition appeared. Notably, Lambert derived his perspective from Martin G. Christopher's Logistics, The Strategic Issue (London: Chapman and Hall, 1992), which explicitly references the 1982 publication by Oliver and Webber: (Lambert, 2004, p. 2)

1.5.2. 1990s: Expansion and Strategic Approach

During the 1990s, the SCM concept was further expanded and redefined through a strategic lens. A notable contribution in this period came from Lambert, Cooper, and Pagh in their 1998 article "Supply Chain Management: Implementation Issues and Research Opportunities" published in the International Journal of Logistics Management. Their work emphasized:

- the definition of SCM as integration of key business processes from key users to stakeholders. (Lambert, 1998, p. 1)
- that successful SCM requires seamless integration of all members within the supply chain. (Lambert, 1998, p. 15)
- that SCM consists of 3 elements: supply chain network, supply chain business processes and management components (Lambert, 1998, p. 15)

This period marked a significant shift in thinking, as researchers and practitioners alike began to view SCM not just as a set of operational practices but as a core component of strategic management.

1.5.3. Early 2000s: Comprehensive Definitions and Practical Applications

The early 2000s witnessed further refinement of the SCM concept, with several key publications offering comprehensive definitions and practical frameworks:

Mentzer et al. (2001)

Another contribution was done in "Defining Supply Chain Management" published in Publication: Journal of Business Logistics. This seminal article provided one of the most

widely recognized definitions of SCM at their time (Menzter et al., 2001, p. 6) and by doing this Mentzer and his coauthors gave their definition of SCM "as the systemic, strategic coordination of the traditional business functions and the tactics across these business functions within a particular company and across businesses within the supply chain, for the purposes of improving the long-term performance of the individual companies and the supply chain as a whole." (Menzter et al., 2001, p. 18)

1.6. OMS and WMS

In scientific literature, there are no precise indications of when the term Warehouse Management System (WMS) was first introduced or how it evolved over time. As a result, it is difficult to provide a concise history of its development.

According to Oracle, a Warehouse Management System (WMS) is a software solution designed to efficiently manage and optimize warehouse operations, including receiving, storage, picking, packing, and shipping of goods.[1].

According to Nynke Faber, René B.M. de Koster, and Steef L. van de Velde, a Warehouse Management System (WMS) is a system that "provides, stores, and reports the information necessary to efficiently manage the flow of products within a warehouse, from the time of receipt to the time of shipping". (Faber et al., 2002, p. 382)

Another definition was given by Susanna Patton and Crispin Coombs in their article "Factors Affecting the Level of Success of Warehouse Management Systems". They describe WMS as a system that "monitors the goods-in, put-away, rotation, pick and dispatch of all stock units held within a warehouse." (Patton & Coombs 2009).

So, in general WMS can be described as a system which manages the lifecycle of goods within a warehouse. It's main definition to make this process easier and more efficient and moreover make it certain that all goods will be picked up and all of them will be shipped in the right time.

Concerning OMS there is no one concrete definition for this system.

[1] https://www.oracle.com/cis/scm/logistics/warehouse-management/what-is-warehouse-management/ (25.03.2025)

The OMS system is a system which manages the lifecycle of an order. This definition is given on the IBM Site.[2]

Another explanation of OMS gives Microsoft's site. System that automates tracking the number of sales, orders, inventory, and fulfillment, and this sounds more like a mix of SCM system, and a definition of OMS given by IBM.[3]

Oracle gives an OMS definition, which is similar to the IBM's definition: it is something, which "supports all the stages in your company's sales process – from order creation through delivery and even returns."[4]

SubKit gives OMS the same definition as Microsoft. OMS is a "digital system that tracks sales, orders, inventory, and fulfillment, ensuring that all aspects of the business run smoothly".[5]

Clearcode CC gives another definition for OMS. It says, that "An order management system (OMS) for publishers handles the whole sales process to meet their customers' demands and grow their company's revenue." And this definition sounds like a mix of ERP system and an OMS system described by IBM.[6]

OneMoneyWay describes OMS as "system (OMS) is a central hub that connects different stages of order processing, from when an order is placed until it is fulfilled." And again this definition may be rephrased as a system which manages order's lifecycle.[7]

---

[2] https://www.ibm.com/think/topics/order-management (25.03.2025)

[3] https://www.microsoft.com/en-us/dynamics-365/topics/intelligent-order-management/order-management-system-oms (25.03.2025)

[4] https://www.netsuite.com/portal/resource/articles/erp/what-is-oms.shtml (25.03.2025)

[5] https://learn.subkit.com/order-management-system-oms-guide-to-supply-chain-optimization-for-e-commerce (25.03.2025)

[6] https://clearcode.cc/blog/what-is-order-management-system/ (25.03.2025)

[7] https://onemoneyway.com/en/dictionary/order-management-system/ (25.03.2025)

There is also another definition of OMS given by yango-tech.vercel.app. “An Order Management System (OMS) is software that helps businesses efficiently manage orders from multiple sales channels. Whether the order originates from a brick-and-mortar store, an ecommerce platform, or a mobile application, the OMS consolidates and processes them seamlessly.”. This definition of OMS is similar to IBM’s definition, but it develops it further. It says that even if a company has different apps for customers the OMS system track orders from all these apps.[8]

Another interesting definition is given by Commerce-Docs website. “Order Management System (OMS) is a flexible and affordable solution for managing, selling, and fulfilling inventory from any sales channel.”. And this definition is mix of Microsoft definition with definition given by “yango-tech.vercel.app” as Microsoft also was speaking about fulfillment an inventory and a second source mentions about controlling different salesmen sources.[9]

Concluding all these definitions the following can be said. OMS can’t be described as ERP because it means not to develop enterprise but manage the lifecycle of an order. It also can’t be classified as CRM, because of the definition of CRM to develop close relationships with customers. It can be described as SCM because it doesn’t monitor supplies. It also can’t be classified as HRM, because it doesn’t manage people within a company, and finally, it can’t be classified as WMS, because WMS software is developed for warehouses, not for businesses with orders. And it means that OMS is indeed a different system with different purposes.

All these articles give a slightly different definition of what the OMS system is, and for example the definition of OMS, given by Microsoft is in some way similar to ERP system, but there is one feature, which is included into all articles – OMS is the system, which manages the lifecycle of an order. With this information, it can be concluded that the OMS system is a system which manages the order’s lifecycle, from its start to its

[8] https://yango-tech.vercel.app/blog/order-management-system-oms-features-benefits (25.03.2025)

[9] https://commerce-docs.github.io/oms-documentation-archive/getting-started/ (25.03.2025)

completion. It stores data of orders, which includes data of clients and products alongside data of operators who fulfil this order.

2. ANALYSIS

2.1. Requirements for the future system and its type

The car wash management system must handle the entire lifecycle of a car wash order, ensuring efficient management of vehicle servicing and customer interactions. The system should support various scenarios and requirements outlined below:

2.1.1. Managing Customers with Multiple Vehicles

A single customer may own multiple cars (e.g., a family with two vehicles).

The system must allow to register a customer, manage and schedule car washes for each of customers vehicles separately.

The booking history should be linked to the specific vehicle rather than just the customer.

The ability to work with clients who have multiple contact details of varying importance (personal contact information and contact information linked to a company).

2.1.2. Handling Vehicles with Multiple Owners

A vehicle may have multiple owners, such as in a family where both adults share a car.

If a second owner of a vehicle, who has not yet been registered in the system, arrives at the company, the system must record the new client's data and associate it with the existing vehicle in the database without removing the link to the previous owner.

The system must support access to data for all registered owners of a vehicle.

2.1.3. Support for Corporate-Owned Vehicles

Support for vehicles which may be registered under a company rather than an individual.

The system must allow ordering for multiple representatives from the same company.

If a company representative arrives to place an order with a vehicle owned by the company, they should be able to register the order both under the company's and their own name.

If a company representative arrives to place an order with a vehicle that does not belong to the company, the system must allow them to register the order under their own name.

2.1.4. Multiple Contact Numbers for a Single Customer

Some customers may have both a personal and a business phone number (e.g., employees using a company vehicle) that is why the system must:

- Allow users to choose which phone number to use when booking a service.
- Contact preferences should be configurable, allowing communication via the preferred phone number.

2.1.5. Environmental impact tracking

The system must document the environmental impact of each car wash.

The existing classification system categorizes washes into three pollution levels:

- High pollution
- Medium pollution
- Low pollution

The new system must integrate with the company's current pollution classification mechanism and generate necessary reports for regulatory and internal use.

2.1.6. Type of the Information System

This information system is designed solely to manage the lifecycle of an order. The following points should be noted:

Not an ERP System

The system does not manage the production processes of the enterprise, nor does it contribute to improving the business model, which excludes it from being classified as an ERP system.

Not a CRM System

There is no direct interaction with customers, nor does the system include functions aimed at managing customer relationships, so it cannot be considered a CRM system.

Not an HRM System

Since the system does not manage human resources, it cannot be categorized as a Human Resource Management (HRM) system.

Not an SCM or WMS

The system does not manage warehouses or supplies, meaning it does not fit the definitions of Supply Chain Management (SCM) or Warehouse Management Systems (WMS).

Thus, given that the primary function of this system is to manage the order lifecycle, it can be classified solely as an **OMS (Order Management System)**.

2.2. Analysis of competing solutions

The following chapter will give an overview of companies which already have car washing business in Narva. This is done to better understand what type of the IS is better suited for this business. The research shows that there are already 9 companies in Narva which provide car washing services.

Unfortunately, it was not possible to get in touch with the companies. For this reason, the information system description will be based on the service descriptions available on their websites, as well as on the applications provided by some of the companies.

1. SuperStiil Autopesula ja Rehvitööd (Kangelaste prospekt 8c, Narva) – no website, but there is a page on Facebook and Instagram. There are no sections for customer interaction. The type of information system can't be determined.[10] [11]
2. Автомойка Самообслуживание (A. Daumani tänav 2e, Narva) – no website, no social media pages, the type of information system is unknown.
3. GENOME AUTOPESULA (Rahu tänav 7a, Narva) – it is not possible to determine exactly what type of information system is used, because there is only price list on the

[10] https://www.facebook.com/superstiil/ (25.03.2025)

[11] https://www.instagram.com/superstiil_narva/ (25.03.2025)

webpage and no other information is presented. There are also pages on Facebook, Instagram and VK.[12] [13] [14] [15]

4. Circle K | Autopesu (Eesti) – has a CRM application where the customer can order the service and contact support. The application consists of five sections: Home, Stores, Extra, Deals, and Account. On the Home page, users can pay for fuel purchased at the gas station and select a subscription for a specific type of car wash, along with viewing advertisements and an offer to purchase a customer loyalty card. The Deals section was empty at the time of review. In the Extra section, users can purchase services available exclusively to clients. The Account section allows users to manage their personal information and contact customer support by sending a message. It also has a website.[16] [17]
5. Nutipesu – has a CRM branch of the main system, presented as a contact form where customers can ask questions or make suggestions. IS type of the main system can't be determined.[18]
6. Mündipesula – has a CRM-oriented branch of the main system with an online customer support form. Additionally, there are other ways to contact the company through their website.[19] Type of the main IS can't be determined
7. Jazz Pesula – has a CRM-oriented application. The application consists of five sections. In the Home section, users can view nearby car washes, order a customer card, visit the company's social media pages, make a phone call (although this feature did not work on the author's Google Pixel 7 due to an error), view current

[12] https://genome.ee/ (25.03.2025)

[13] https://www.instagram.com/genome.ee/ (25.03.2025)

[14] https://www.facebook.com/genome.ee (25.03.2025)

[15] https://vk.com/genome_ee (25.03.2025)

[16] https://www.circlek.ee/extra (25.03.2025)

[17] https://play.google.com/store/apps/details/Circle_K?id=com.circlekeurope.android&hl=ru&pli=1 (25.03.2025)

[18] https://nutipesu.ee/ (25.03.2025)

[19] https://myndipesula.eu/ru/ (25.03.2025)

promotions, and check the weather. The Locations section provides a list of all their car washes. In the Map section, users can view their current location on a map. The Promotions section displays the ongoing offers. There are no sections available for placing a car wash order or managing user data. It also has a website. [20] [21]

8. TerminalOil – has a CRM branch. Recently it changed domain name and now if open a link with an old web address it will route you to the new website.[22] [23]
9. Volvopesula – (Spordi tänav 6, Narva) – no information available except for the address and phone number.

The following research gives these results. Four companies out of nine do not have any website or don't provide information about the services which they provide. Five companies out of nine have CRM oriented applications, which they provide through Google Store. While there is no more exact information about the IS of these companies, we may conclude that for a successful business only the OMS system may not be enough.

2.3. Analysis of software solutions

The following chapter research already existing software solutions to make it clear, if the development of the new OMS system is justified, or whether it is better to use one of these existing solutions. The solutions through the first two pages in the Google search results were reviewed.

Here was the same situation as with companies located in Narva. Unfortunately, it was not possible to get in touch with the companies. For this reason, the information system description will be based on the opportunities descriptions available on their websites

1. Car Wash Operation and Management Software – There is no exact description of what is included in the software package. Only advertisements are available. Based on the information provided on the website, it can be concluded that this system is more of an ERP system rather than an OMS (which needs to be developed), as the

[20] https://www.jazzpesulad.ee/pesutanav (25.03.2025)

[21] https://play.google.com/store/apps/details?id=ee.jazzpesulad.app (25.03.2025)

[22] https://terminaloil.ee/ (25.03.2025)

[23] https://terminalenergia.ee/autopesu/ (25.03.2025)

system not only manages the order lifecycle (in this case, car washing) but also other production processes.[24]

2. Pulsewash – Based on the system's described capabilities, it is an ERP system targeted at car washes located at gas stations, as it includes tools for kiosk management in addition to car wash functionality. The system is more focused on self-service car washing, which does not align with the project's goals.[25]
3. BKF CarWash – Offers multiple information systems, including an OMS for car wash management. However it doesn't have any description of whether it is possible to manage owners who own a few cars or is it possible to add several personal contacts for a person. [26]
4. Dibo – A more ERP-oriented system, as it includes functions beyond just order lifecycle management.[27]
5. Rinsed – A CRM-oriented system, as stated directly on the website.[28]
6. Sciwash – Based on the description, it is an OMS system. It allows management of multiple branches and supports payment via bank terminals. There is no information on whether the system allows managing customers with multiple contact details and multiple vehicles. There is also no information on whether it allows managing vehicles with multiple owners.[29]
7. Car wash booking system – Best software & Free app – Based on the description, it is an OMS system. There is no information on whether it allows managing multiple branches. There is also no information on whether it allows managing customers

[24] https://www.washup.solutions/ (25.03.2025)

[25] https://www.pulsewash.com.au/ (25.03.2025)

[26] https://bkfcarwash.eu/ (25.03.2025)

[27] https://www.dibo.com/en/vehicle-washing-systems/self-service-car-wash (25.03.2025)

[28] https://www.rinsed.com/the-car-wash-crm (25.03.2025)

[29] http://sciwash.com/ (25.03.2025)

with multiple contact details and multiple vehicles or managing vehicles with multiple owners.[30]

8. Mcarw – Based on the description, it is a mix of ERP, HRM, and SCM systems, as it includes functionality for managing not only the order lifecycle.[31]
9. Adriateh – A modified OMS system, as it includes marketing management functionality.[32]
10. DRB – A CRM system, as it includes functionality for customer interactions.[33]
11. Serpents – There is no information on whether it allows managing multiple branches. There is also no information on whether it allows managing customers with multiple contact details and multiple vehicles or managing vehicles with multiple owners.[34]
12. am8ze – A mix of ERP, CRM, and SCM systems.[35]
13. Moiboo – An OMS system. There is no information on whether it allows managing multiple branches. There is also no information on whether it allows managing customers with multiple contact details and multiple vehicles or managing vehicles with multiple owners.[36]
14. Odoo – A CRM-oriented system based on the description, as it includes customer interaction functionality.[37]

---

[30] https://www.booklux.com/en/car-wash-booking-system#trusted_clients (25.03.2025)

[31] https://mcarw.com/ (25.03.2025)

[32] https://adriateh.com/car-wash-accessories/ready2wash/ (25.03.2025)

[33] https://drb.com/tunnel_solutions/point-of-sale/washify (25.03.2025)

[34] https://www.serpentcs.com/products/car-wash-management-system (25.03.2025)

[35] https://am8ze.com/car-wash-management/ (25.03.2025)

[36] https://www.moiboo.com/car-wash-management-software.php (25.03.2025)

[37] https://apps.odoo.com/apps/modules/15.0/car_washing_management (25.03.2025)

15. Bsquare – An OMS system. There is no information on whether it allows managing multiple branches. There is also no information on whether it allows managing customers with multiple contact details and multiple vehicles or managing vehicles with multiple owners.[38]
16. Cisworld – An ERP system based on the description, as it includes functionality for managing all business processes.[39]

The following results of research were given. Four software solutions are ERP oriented software which main target is to develop business processes of company. And this means that it is not suited for the described purpose. Five software solutions are OMS, which is the IS type what is needed, but it does not meet the requirements or has not enough additional information about the system and its capabilities. Two software solutions are CRM systems, and this does not meet the requirements. And the latest software solutions are mixes of different IS, which do not meet the requirements that the IS should be an OMS system. With this information given it can be concluded that none of these solutions meet the requirements for the system to be developed.

[38] https://www.bsquare.in/car-wash-management-software.html (25.03.2025)

[39] https://www.cisworld.lk/car-wash-management-system (25.03.2025)

3. DEVELOPMENT

Methods for implementing an independent solution

- Programming Language: C#
- User Interface: Avalonia UI
- Database: Microsoft SQL Data Server, SSMS
- IDE: Visual Studio

3.1. Preparations

3.1.1. Database

The first step should have been the development of the database. Based on the client's requirements, the application must be able to collaborate with customers who may own multiple cars, as well as with cars that may have multiple owners. Additionally, the client requested that the database be manageable, if possible, from a Windows operating system. For this reason, Microsoft SQL was chosen as the DBMS software. Therefore, the database is a object-oriented database.

3.1.2. Relationships

Next, the discussion will turn to the organization of relationships in the database. The three most prevalent types of these connections are "One-to-One", "One-to-Many", and "Many-to-Many".

3.1.3. One-to-one

The potential issues that could arise in each type of relationship will now be examined. Starting with the "One-to-One" relationship type. A customer comes to the company in their car. After an undefined period of time, the customer's financial situation improves, allowing them to purchase another car, while still wanting to keep their old one. In this case, if a "One-to-One" relationship is implemented in the database, it will be impossible to establish a new relationship between the existing customer and the new car, as it would conflict with the primary key constraint.

3.1.4. One-to-Many

Two potential issues that may arise when implementing a “One-to-Many” relationship model are now considered. The first scenario is where the main linking element in the database is the customer. A client arrives at the company in their car. Their data is entered into the database. Sometime later, the client’s wife arrives at the company in the same car. In this case, it may be possible to enter the new client’s data into the database but linking her and the car to the service will no longer be possible, because the car has already been associated with her husband. Deleting the existing relationship where the car is linked to the husband would be incorrect, as both spouses are in fact co-owners of the car.

Similar issues may also arise in other scenarios involving a client who owns two vehicles – one registered to a company and the other being their personal car. These complications can occur in both “One-to-One” and “One-to-Many” relationship models. In either case, a situation may emerge where the client initially visits using their personal vehicle and provides personal contact information. Later, the same client may return in a company-owned vehicle, this time representing a legal entity. In such a scenario, the database would not allow the addition of new contact information due to key constraints.

A comparable issue would also occur in the reverse situation, where the vehicle serves as the primary linking entity. If the client purchases a second vehicle, it would be impossible to create a new association with the new vehicle, as it would conflict with the existing one.

3.1.5. Many-to-many – is the only appropriated relationship

To summarize the selection of relationship types in the database: the “One-to-One” model does not meet the project requirements, as a single client can own multiple vehicles. The “One-to-Many” relationship is also unsuitable. If the client is treated as the main entity capable of owning multiple vehicles, it contradicts the requirement that a vehicle can have multiple owners. The same problem arises when the vehicle is considered the primary linking element.

Based on all the above, it becomes clear that the only appropriate relationship model for the database is the “Many-to-Many” model.

The prototype of the database, including the main tables and their data types, was developed by students Sergei Markov, Danil Gubar, Yuriy Krasnobokiy, and Vladimir Galitski as part of the "Databases" course. The current version of the database used in the application was developed by Sergey Markov.

3.1.6. IDE & Programming language

The reason Avalonia UI (which is based on C#) was chosen as the programming framework lies in the client's request for a cross-platform application that could run at minimum on both Windows and Linux. Avalonia UI enables the development of applications that work on Windows, Linux, and even Android. However, it is important to note that this framework is still relatively new and may not be the most optimal choice for developing Android applications.

The chosen integrated development environment (IDE) for the project was the free version of Microsoft Visual Studio.

3.2. Main application components

3.2.1. Consoles

The application is designed to include two main consoles: the Administrator Console and the Operator Console. The Administrator Console is the part of the program intended for administrative tasks, such as modifying operator data, managing product-related information, updating prices, and more. The Operator Console is the part of the application dedicated to manage order's lifecycle. The author of this work is responsible for the development of the Operator Console.

Before proceeding further, it is necessary to explain several key terms and their meanings within the context of Avalonia UI

The application utilizes the following components: Page, View, and Control.

3.2.2. Pages

Page represents the current state of the application, occupying the entire screen. It can be compared to a webpage. Like a webpage, it requires a view (analogous to a PHP template), which in Avalonia is implemented using XAML, as well as a view model that handles all logic and user interaction on the page.

3.2.3. Screen

A screen is a view inside view. It has a fixed-height and fixed-width section of the page. It is used to simplify future modifications of the application. Unlike panels, their size and position remain unchanged, regardless of changes in dynamic content. This makes it a stable structure within the layout.

3.2.4. Controls

Controls refer to dynamic UI components that appear as needed. They can be thought of as miniature views. Like views, controls do not require a separate view model to function. The typical usage involves embedding a `<TransitioningContentControl>` element within a page or view. This element is bound to a variable (via data binding) inside the page's view model, which manages the data and user interactions for that page.

When it is necessary to switch from one control to another, the bound variable is assigned a new value. For example, if the variable is named CurrentPage, switching to a new control might look like:

```
CurrentPage = new VehicleScreen();
```

Here, "`CurrentPage`" is the name of the variable in the view model, and "`VehicleScreen()`" is the name of the control.

Each control has an automatically generated companion file with the same name but with the extension ".axaml.cs", also as in view. These files are necessary because certain user interactions—such as mouse clicks, hover events, and similar actions—are either not directly accessible from the view model or would require significantly more complex and verbose code to implement there. These code-behind files allow such events to be handled more efficiently and cleanly.

3.2.5. Records

A record is a data structure consisting of a fixed number of components, known as fields. Each field can have its own data type. It can be compared to a List; however, unlike lists, records are immutable once created. One of the key advantages of using records lies in encapsulation – they are lightweight and are processed more efficiently. In this application, records are used to avoid repeated database queries by storing and

retrieving values directly from previously saved data. This approach significantly improves performance and reduces the risk of overloading the server with multiple database requests.

3.2.6. Images

SVG refers to vector images. Since raster images tend to distort when window size increases, it was decided to use SVG format to maintain image quality at any scale. Unfortunately, SVG support in Avalonia is currently limited and may not function correctly in all scenarios – even when the code is written properly. As a result, images in other formats are also used when necessary.

PNG, JPG, and JPEG are raster image formats used as alternatives when SVG is not supported or does not display correctly.

3.2.7. Layout

There are two primary ways to structure page layouts in Avalonia UI to ensure that elements maintain their positions regardless of dynamic content or window size changes. The first method is using Grid, and the second is using View. The decision was made to use View instead of Grid, as it offers a more flexible development approach. In case future modifications are needed, Views allow for easier and quicker layout adjustments. Grid, by contrast, is better suited for static layouts with predefined structures and requires significantly more effort to modify once implemented.

3.3. Controls

All Controls used in this application are in the “Controls” folder, which is divided into two subfolders: “Kassa” and “ManagerConsole”. Since the author of this work was responsible for developing the Kassa branch, the discussion will now focus on the controls found within the Kassa folder.

3.3.1. CalendarScreen

`CalendarScreen` is a control created with future development in mind. The client requested that the application includes a feature for searching orders by date, as well as a calendar page. However, since the client has not yet fully defined how this functionality should look or behave, the control was left in a preliminary state. It is intended

to serve as the foundation for a dedicated calendar control that can later be refined and completed based on further requirements.

3.3.2. VehicleScreen

VehicleScreen is a control designed as a form where the operator can input a vehicle's license plate number for subsequent database searches. It includes a search field and three buttons: "Private", "Empty", and "Company". By pressing one of these buttons, the operator sets the significance status of the order, whether the order is being created for a private individual or a legal entity.

- Clicking the "Private" button indicates that the order is being placed for a private individual.
- Clicking the "Company" button means the order is being created on behalf of a business.
- The "Empty" button indicates that the client does not wish to be registered in the company's client database or has other reasons for avoiding registration. In such cases, order data will not be saved to the database.

At this stage, this structure was implemented due to the client's current uncertainty about what the final version of the application should look like. To allow the program to remember the significance of the order, both the "Private" and "Company" buttons are assigned corresponding string parameters ("Private" and "Company"). These parameters are passed into the function `ShowClientsConnectedWithVehicle`. Depending on the received parameter, the application determines the order's status and the subsequent program behavior – such as whether to display a list of companies associated with the entered vehicle.

Both the "Private" and "Company" buttons remain disabled until the user enters a value into the search field. If the search field becomes empty again, the buttons are automatically disabled once more. Each button's enabled state is controlled by a corresponding binding: `PrivateButtonEnabled` and `CompanyButtonEnabled`. The search field itself, implemented as a `TextBox`, uses a dynamic binding called `CarNumber` to track its current value.

More detailed information about the methods involved can be found in the section 3.6.

3.3.3. ClientConnectedWithVehicle

`ClientConnectedWithVehicle` – is a control designed to display information about all clients associated with a vehicle that has arrived at the car wash facility. The control features a DataGrid, which presents the first name, last name, and phone number of each client linked to the given vehicle.

The DataGrid was chosen over a regular grid because it dynamically adjusts based on the list of data it receives. It automatically generates the required number of rows to match the number of items in the list and fills each cell with the appropriate data. For this automatic population to work, the data list must be composed of either arrays or records and must be bound to the DataGrid as a variable.

In addition to the DataGrid, the control contains three buttons: “Back”, “Next”, and “Another Client”. The “Another Client” button is intended for situations where the person visiting the company is not listed among the clients associated with the vehicle. When this button is clicked, the application transitions to the next control.

Each row in DataGrid displays a client’s phone number, first name, and last name. If a client has multiple phone numbers, the client will appear multiple times in the list – once for each number. This implementation satisfies the requirement that the application must handle clients with multiple contact entries of varying significance.

The “Next” button is initially disabled by default and becomes enabled only after the operator selects a client from the list. This behavior is controlled through the binding `VehicleClientsScreenNextButtonEnabled`, which is linked to the `IsEnabled` property of the button. The methods responsible for client selection and enabling the “Next” button are detailed in the subsection 3.6.

Since the appearance of the buttons – such as text color and icon – changes based on whether the operator hovers over them or not, each button’s text color and image path are also tied to specific bindings.

3.3.4. ClientPhoneSearch

`ClientPhoneSearch` – is a control designed to search for clients using their phone numbers. This screen appears when a vehicle’s license plate number is not found in the database. It can also be triggered from the `ClientConnectedWithVehicle`

control, where the operator can click the “Another Client” button if a different client arrives with a vehicle that is already registered in the system.

The control includes a search text box and two buttons: “Back” and “Search.” Similar to the behavior in the previous control, the Search button remains disabled until the operator enters a value into the search field. If the input is cleared, the button becomes disabled again.

To track the state of the input field, the text in the TextBox is bound to the `ClientPhone` property. This allows the application to remember the operator’s input, so if they navigate away and return to this control, the previously entered value will still be displayed.

If a matching client is found, the application automatically creates a link between the new vehicle and the existing client. Afterward, the operator is redirected to the next control – `ClientConnectedWithVehicle`.

3.3.5. NewClient

`NewClient` – is a control created for registering new client information in the database. It appears when the phone number search does not return any results. The control consists of four TextBoxes, each labeled with a corresponding TextBlock. In addition, it includes two buttons: Back and Save. The Back button returns the operator to the `ClientPhoneSearch` control. This navigation is handled through the Command parameter of the button. As with previous controls, the Save button in the `NewClient` control remains disabled until the operator fills in all required fields. Once the Save button is pressed, the new client’s information is added to the database, a link is created between the client and the vehicle associated with the order, and the operator is then redirected to the `ClientConnectedWithVehicle` control, where the updated client list, now including the new client, is displayed.

3.3.6. CompanyConnectedWithVehicle

`CompanyConnectedWithVehicle` is a control created to display company information linked to a vehicle. The operator sees this control only if the button Company was selected in the `VehicleScreen` control. This control consists of a DataGrid and three buttons: Back, Another Company, and Next. The DataGrid includes three columns: Company Name, Company Address, and Is Active.

Unlike the ClientConnectedWithVehicle control, the Next button in this control is always enabled, since only one company can be associated with a vehicle at a time – vehicles cannot be registered to multiple companies. The button "Another Company" is used when ownership of the vehicle has changed, and the operator needs to assign a new company to it.

3.3.7. FindCompanyName

FindCompanyName is a control created for searching companies in the database by their name. It appears when the operator clicks the "Another Company" button in the CompanyConnectedWithVehicle control. This control is structured similarly to ClientPhoneSearch, containing a search field and two buttons: Back and Search. The button Back navigates the operator back to the CompanyConnectedWithVehicle control.

As with ClientPhoneSearch, the Search button is disabled until the operator enters text into the search field. If the company is found in the database, the operator is redirected back to the CompanyConnectedWithVehicle control. If not, the next control is triggered.

3.3.8. NewCompany

NewCompany is a control created for entering data for a new company that hasn't yet been registered in the system. It appears when a company search yields no results. The control includes three TextBoxes, each with a corresponding TextBlock, and two buttons: Back and Save. As with the NewClient control, the Save button remains disabled until all fields are filled in. Pressing the Back button returns the operator to the FindCompanyName control, while pressing Save registers the new company in the database and links it to the vehicle associated with the order.

3.3.9. ServicesListScreen

ServicesListScreen is a control designed to display a list of services offered by the client's company. These can include types of car washes and other services. The control contains a DataGrid and a Back button. It appears by default when the operator presses the Next button in CompanyConnectedWithVehicle control. It can also be accessed from the button Services located in the InnerTransactionPanel view.

The DataGrid consists of four columns: Trade Unit, Price, Start Time, and End Time. The Trade Unit shows the service name, Price indicates the cost and Start Time with End Time provide estimated timeframes for service execution. When a service is selected, it is added to the shopping cart. This functionality is handled by the `OnSelectionChanged` method assigned to the `SelectionChanged` parameter. Database stores the duration of each services and function, which get the list gets a current time, and then adds duration to the current time.

3.3.10. GoodsListScreen

`GoodsListScreen` is a control created to display a list of goods sold by the client's company. It is similar in structure to `ServicesListScreen`, featuring a DataGrid and a Back button. When a product is selected via a click in the DataGrid, it is automatically added to the cart through the `OnSelectionChanged` method bound to the `SelectionChanged` parameter. The DataGrid includes two columns: Trade Unit and Price.

3.3.11. ShoppingCart

ShoppingCart was created to show a list of items and products selected by the client. It is nearly identical to the GoodsListScreen, also featuring a DataGrid and a Back button. However, unlike GoodsListScreen, the DataGrid here is bound to a different variable: ItemsInShoppingCartList.

3.3.12. OrderScreen

OrderScreen is a control created to display the data of an active order being processed in the company. Additionally, it can be used in the future as a general-purpose control for displaying order details. It includes a Grid and two DataGrids. The Grid displays client information for the current order. Since this section handles only one entity, a Grid is used instead of a DataGrid.

The two DataGrids display the list of goods and services ordered by the client. If only goods were purchased, only the goods table is shown. If only services were ordered, then only the services table appears. A button "Order Completed" is also present, and pressing it marks the order as fulfilled.

3.3.13. WarningOrderAlreadyExists

WarningOrderAlreadyExists is a control created to display a warning indicating that an order has already been registered for the given vehicle.

3.4. .axaml.cs files functions

As mentioned earlier, it was not possible to handle events such as hovering the cursor over a button or changing its background and text color through the page's ViewModel alone. Avalonia does not allow this to be done directly. To implement such functionality, it was necessary to create two separate functions for handling the same event on the same object. One of these functions is located in the ".axaml.cs" file and is responsible for tracking the cursor's behavior, while the other resides in the page's ViewModel and processes the event.

The function in the ".axaml.cs" file calls the corresponding function in the ViewModel. The answer to the question "Why can't this be done solely in the .axaml.cs file or in the ViewModel?" is quite simple: ".axaml.cs" files do not have the required functionality on their own, and this particular property in XAML works only through the ".axaml.cs" code-behind.

For this reason, such dual-function setups are necessary in the application. Additionally, since all these functions follow a similar pattern, they will be grouped into classes in this description for the sake of compactness and readability.

3.4.1. Cursor hovering functions

First of all, in order to implement such functionality, objects in XAML have a property called PointerEntered. This property must be linked to a method inside the .axaml.cs file. For example, if there is a button named Back, its properties should include `PointerEntered = "VehicleClientBackButtonCursorEntered";`

Then, for the .axaml.cs file to recognize the button, it must be assigned a name using the Name property. This would look as follows:

```
Name = "VehicleClientBackButton";
```

The focus now shifts to the function within the .axaml.cs file. This function is defined by a name and an argument:

VehicleClientBackButtonCursorEntered(object? sender, PointerEventArgs e);

The PointerEventArgs e argument indicates that the incoming value is directly related to cursor actions, while object? sender suggests that the function will interact with an object. The question mark denotes that this could be any object.

Next, the function needs to locate the specific object. To do this, a variable is created:

```
var VehicleClientBackButton = this.FindControl<Button>("VehicleClientBackButton");
```

Image 1. Variable VehicleClientBackButton

Here, FindControl is the method used to search for the object, <Button> specifies that it should search for a button, and VehicleClientBackButton tells it which button to find by name.

Since the button may be disabled, conditional if-else statements are used. In both cases, the function calls another method that takes a single bool argument. If the button is active, the method is called with true when the cursor enters and with false when the cursor exits. In this case, the call would look like:

```
if (DataContext is MainKassaScreenViewModel viewModel)
{
    viewModel.VehicleClientBackButtonCursor(true);
}
```

Image 2. Addressing to viewModel

For the relevant object – in this case, the button – the text color, image path, and background color are all bound to corresponding variables. This function then updates those values based on the received argument.

Since the button may be disabled, conditional if-else statements are used. In both cases, the function calls another method that takes a single bool argument. If the button is active, the method is called with true when the cursor enters and with false when the cursor exits. In this case, the call would look like:

Accordingly, when the cursor leaves the object, the process is entirely identical to when the cursor enters the object, with the only difference being that a different function is called and the PointerExited parameter is used. As a result, the property in the XAML would be written as follows:

```
PointerExited="VehicleClientBackButtonCursorExited";
```

3.4.2. Methods for selecting an item in a Datagrid

Interaction with the `ViewModel` is handled in the same way as cursor-related actions. To track changes in selection, the `SelectionChanged` property is used. This property calls a function in the `ViewModel` named `SelectClientExecute(object? sender, SelectionChangedEventArgs e)`. Then, to store the selected item, the data is saved in the form of a `Record`. For this, a variable is created from the selected item using record format.

```
var selectedClient = selectedItem as CommunicationClientRecords;
```

This variable is then assigned to the corresponding property.

3.5. Functions that work with the database

All functions that interact with the database are located in files found in the path `Services\DBMSServices\`. If a function is intended to work with the `Branches` table, it will be located within the `Branches` file. In Avalonia, there are several ways to work with a database. Two common approaches include: connecting to the server where the database is hosted and writing custom SQL queries along with a custom method to handle those queries, or—if an object-oriented database is being used—utilizing the internal language designed for database operations.

Initially, the project used the first approach involving direct SQL queries. However, since the database in this project is object-oriented, a decision was later made to switch to Avalonia's internal data-handling mechanisms, better suited for working with such a structure.

Example of adding a new record to the `Vehicle` table and creating a link with an already existing customer.

```
public static VehiclesRecord? FindVehicleRecord(string licence)
{
    VehiclesRecord returnResult = null;

    using var db = new DbmsService();

    var foundVehicle = db.Vehicles
        .Where(v=> v.Licence.Equals(licence))
        .Include(vehicle => vehicle.Clients)
        .ThenInclude(comm => comm.CommunicationMeans)
        .FirstOrDefault(v => v.Licence == licence);
```

Image 3 - Querry to the database used in Avalonia

3.6. MainkassaViewModel

3.6.1. Commands

`Commands` are used to bind a method from the `ViewModel` to a button in a `View` or `Control`. In order for the View to recognize the command, all commands are assigned the public access modifier. Additionally, to ensure that the behavior of commands updates automatically and is processed asynchronously, `ReactiveCommand` is used. A complete command declaration is written as follows.

`Unit` refers to the return type of the method that the `command` is bound to, while `String` represents the argument that the method receives. In cases where the method does not take any arguments, `Unit` is used instead of `String`. In such a case, the `command` would be written as:

```
public ReactiveCommand<Unit, Unit> NewClientScreen { get; }
```

Once the command is declared, it must be initialized in the constructor and assigned a method to become functional. This is done as follows.:

`ReactiveCommand.Create` is used to initialize a command within the application. An argument is specified only if the command has been defined to accept one. If `Unit` is used, then no argument or curly brackets are included. The `NewClientScreenExecute` refers to the method being assigned to the command. The initialization of a command inside the constructor without arguments would look like this.

3.6.2. Variables

Simple variables

All simple variables in the application look like this one:

Each simple variable is assigned to a `Reactive` status so that they are processed asynchronously. In total, there are more than 70 such variables in this model.

Variables consisted of two components

If it is necessary to track changes to a variable in real time, a two-variable construction like the following is used.

Next, in the `constructor`, it is specified how the application should respond to changes.

In this case, if the company name is empty, a method is called that handles the button color, image path, and text color in the company search control.

3.6.3. Constructor

It contains the declaration of all commands, initial parameters for some variables, and the description of the program's behavior (see Image 10).

3.6.4. AddLoggedInUser

This method checks whether the operator who has just been authorized is already present in the `_loggedInUsers` list. The search is conducted using the operator's Id. If the operator has not yet been added to the list, their data is first added, then the method that handles operator buttons is called, and the operator is redirected to the `VehicleScreen` control.

3.6.5. ExitMenuCommandExecute

Closes the aplication

3.6.6. CheckoutAndExitExecute

It contains a temporary `boolean` variable named `isAnyoneBusy`, which is set to `false` by default. Then, a loop checks whether any of the authorized operators are currently handling an order. If such operators are found, the `isAnyoneBusy` variable

is set to true. If there are no active orders and all operators are free, the application will allow itself to be closed.

3.6.7. NewUserAuthorizationExecute

It switches the application page from the cashier screen to the authorization screen. In the application, page number 0 is always used for the authorization screen.

3.6.8. ShowOrdersScreenExecute

This method is triggered when the “Create New Order” button is pressed. It resets the previously selected client, hides the “Order’s Parameters”view, and then switches to the `VehicleScreen()` control.

3.6.9. SaveNewClientExecute

This is the only method in the cashier module that uses a custom approach for interacting with the database. To implement this, a separate class `RecordBuilder`, had to be created. It is a regular class with an argument (in this case, Record), with setters, but without fields. In this method, work is done with five such RecordBuilders:

- `ClientRecordBuilder`,
- `CommunicationMeanBuilder`,
- `PriceTypeRecordBuilder`,
- `VehicleRecordBuilder`,
- `ClientBonusRecordBuilder`.

After that, a method is called to save all the data. An example of working with setters and `RecordBuilders` is shown in the diagram.

3.6.10. SaveNewCompanyExecute

This method calls a method in the Companies file to save the data of a new company and link it to a vehicle using the necessary arguments. After that, it calls another method, `OrderRouteExecute`.

3.6.11. FindCompanyScreenExecute

Calls control FindCompanyByName.

3.6.12. FindCompanyExecute

The variable `exist` is created by a method in the `Companies` file, and it returns `true` if the company exists and `false` if the company does not exist in the database. If the company exists, a connection is established with the vehicle, and the `OrderRouteExecute` method is called. If the company does not exist, the `NewCompany` control is called. In the company registration number field, the value "nonexistence" is automatically assigned.

3.6.13. Methods that manage button states

These methods change the `IsEnabled` status, text color, and image inside the buttons. The values are changed depending on the argument the method receives. This is all implemented thanks to the global variables in the cash register module.

3.6.14. Cursor hover control methods

These methods change the image of the button and the text color depending on whether the cursor is currently hovering over the button or not. They accept a `bool` argument.

3.6.15. SaveOrderExecute

This method performs the order saving process. It retrieves three GUIDs from the database: the client ID, vehicle ID, and company ID. Then, the first operator from the list of authorized operators is selected, and their ID is taken, followed by the ID of the department where the order was accepted. After gathering all this data, the order is saved, and all irrelevant information for starting the order process becomes invisible again.

3.6.16. SearchClientPhoneExecute

This method searches for the client `id`. If the client is not found, the operator will be redirected to the `NewClient` control. If the client is found, a connection between the client and the vehicle is established by calling the method in `Clients` file, and the `VehicleRoute` function is called. The same happens if a vehicle wasn't found but a client was found.

3.6.17. SelectClientExecute

This method checks which client was selected in the client Grid and then adds that client as the global variable `SelectedClient`.

3.6.18. AddItemToShoppingCartExecute

This method checks which table is currently active and then verifies whether a product or a service has been selected. If either one is selected, it is added to the cart. A separate list, stored in the variable `ItemsInShoppingCartList`, is used to manage the contents of the cart.

3.6.19. SelectServiceExecute & SelectGoodExecute

They operate on the same principle as `SelectClientExecute`, with the only difference being the variables used.

3.6.20. OrderRouteExecute

This method checks which type of client was selected. If a private client was selected, the operator would be immediately redirected to the control for selecting services (the appropriate method is called). If a company client was selected, the operator will be redirected to the `CompanyConnectedWithVehicle` control. If no company is associated with the vehicle, the operator will be redirected to the company search control by company name. A `CompanyVehicle` variable is also created for use in the DataGrid.

3.6.21. ServicesListScreenExecute & GoodsListScreenExecute

In both cases, a list for working with the DataGrid is created by accessing the relevant table. The list of products or services is reset according to the method that was called. A method to generate the client information to be displayed in the right panel is invoked. The corresponding control is called.

3.6.22. VehicleRoute

First, the method checks whether an order has already been placed for the given vehicle. If so, the operator is redirected to a control that warns them an order for this vehicle has already been created and is still incomplete. Next, the method searches the database for the vehicle's number by calling a method located in the Vehicles file. If the vehicle number is not found in the database, the operator is redirected to a control for

searching the client by phone number. However, if the vehicle is found, a loop iterates through all clients associated with the vehicle, and a subsequent loop adds their contact details to a list that will later be used in the corresponding DataGrid.

3.6.23. AddOperatorButtonExecute

Since this method is rather lengthy, only a brief description of its functionality will be provided. This method is responsible for generating the operator's dropdown menu. Because the number of operators involved in the program is unknown in advance, these menus must be generated dynamically, which necessitates implementation through the `ViewModel`.

The process begins with the creation of a `DockPanel`, where the buttons will be placed. The method then iterates through the list of authorized operators, and for each operator, the following is done: an operator button is created along with three potential `DockPanels`, representing three possible options—include in the order, exclude from the order, and end the job.

Since it's not possible to generate buttons within the `ViewModel` that contain `DockPanels` (as can be done in XAML, which Avalonia uses for layout), `DockPanels` are used directly. For each `DockPanel`, an Image variable is created along with the corresponding image path and descriptive text. Although Avalonia supports SVG images through additional packages, these are designed for use with XAML, making it necessary to adopt this workaround

Next, the image sizes, margins, background colors of the `DockPanels`, and their mouse interaction behaviors are defined. To ensure the menu appears above all other elements, a `Popup` is used. Finally, all components are assembled, and actions are assigned so that specific events occur when the user clicks on the corresponding text.

3.6.24. OrderButtonExecute

Here as well, only a general overview will be provided, since the method is lengthy. The image is generated using the same principles as in `AddOperatorButtonExecute`. However, in this case, the element is created as an actual button rather than a panel.

To allow placement of text in the bottom-right corner, a Grid is created. A `TextBlock` is then initialized with all its necessary parameters. After that, both the image and the

text are added to the Grid. An event handler is defined for the button press, which checks whether the lists of selected services or products are empty. Based on this check, the `IsVisible` property is assigned to the corresponding Grid within the `Order` Control. Finally, the generated buttons are added to a `DockPanel`.

3.6.25. ClientInfoStackPanelExecute

This method generates the client information displayed in the order placement view. It initializes the necessary text variables along with their parameters. If the order is placed for a company, additional information about the company is also generated, based on previously stored variables. All of this information is then added to the display panel.

3.6.26. LogoutOperator

It checks whether the operator is currently handling an order. If the operator is not busy, they are removed from the list of authorized operators. If the list becomes empty after removal, the application closes. In all other cases, the `AddOperatorButtonExecute` method is called.

3.6.27. OrderCompleteStatusSaveExecute

The method updates the order record, changing its status in the database to "completed." The operator assigned to the order is released and becomes available to handle new orders. The order is removed from the list of active orders. Methods responsible for generating order and operator buttons are then called, and the operator is redirected to the `VehicleScreen` control.

3.7. Views

This section describes the different Views that are used to generate a checkout page.

3.7.1. CurrentPageView

It represents an empty view with a light brown background. Inside it, a `TransitioningContentControl` is placed to dynamically display the required controls. This is achieved through binding with the CurrentPage.

3.7.2. OrderCheckoutPanel

It contains all the necessary buttons and dropdown lists required for order processing. The interaction principles of the buttons were described earlier in the Control part. This view is always displayed on the right side of the screen.

3.7.3. InnerTransactionPanel

It embeds the two previous views inside it. At the bottom, it features the order menu, implemented as a DockPanel. This menu includes buttons for switching between the service list, product list, and shopping cart controls. Just like the OrderCheckoutPanel, this menu only appears during the final stage of order processing. It takes up approximately 70% of the screen.

3.7.4. KassaLeftPanel

It contains buttons for switching between application modes: order processing mode, annual order overview, calendar, and emergency application shutdown. It is always located on the left side of the screen.

3.7.5. OrderPanel

A view where the dynamically generated buttons are placed. It is always located at the top of the screen.

3.7.6. UserPanel

A view containing dynamically generated operator buttons. It is also located at the top of the screen.

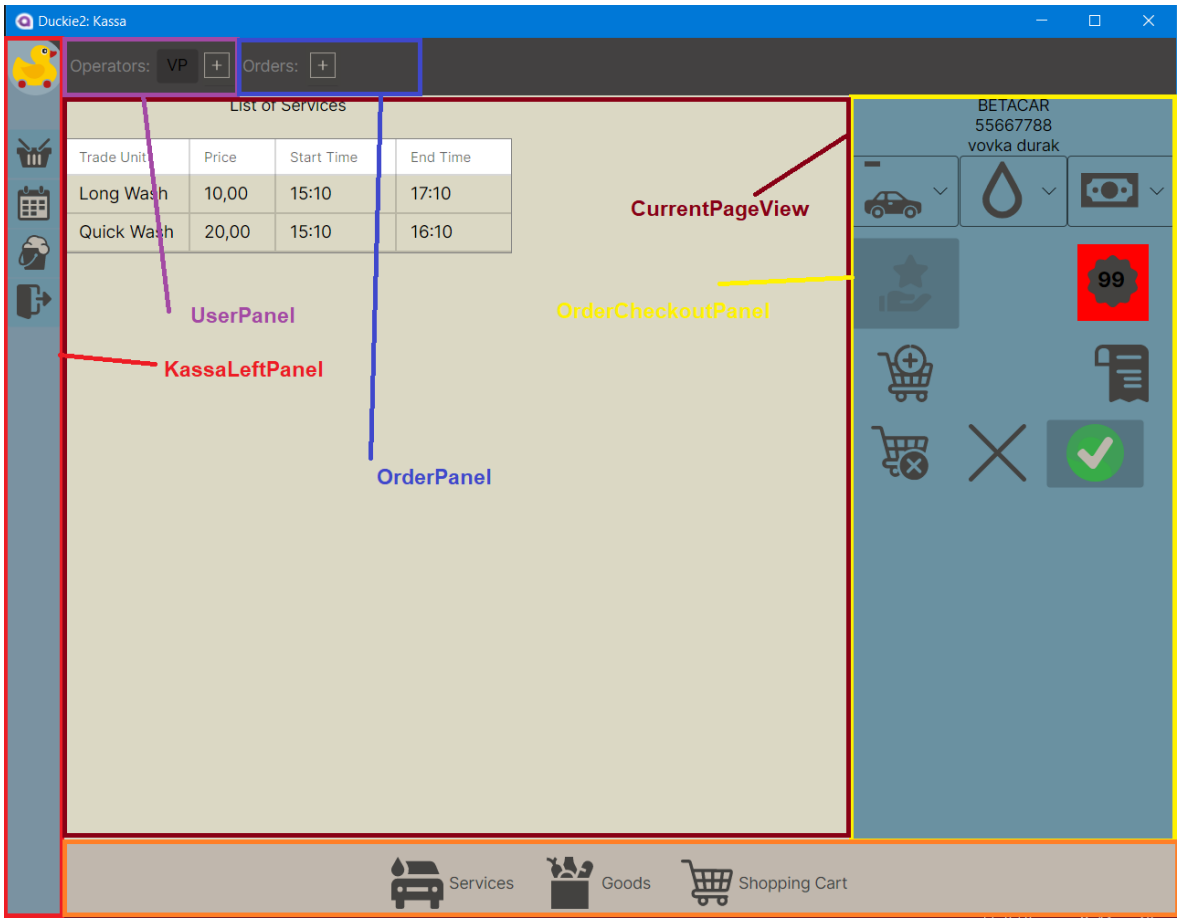

Image 4 - Application

3.8. Application source code

https://github.com/K0rin/Duckie_2_Kassa

CONCLUSION

In conclusion, this thesis has addressed the challenges faced by small car wash businesses in finding software solutions that meet their specific operational needs. The research has shown that each car wash operates under unique business models, and therefore, a one-size-fits-all software approach can't be applied to every business model. Through the analysis of various business information systems such as OMS, CRM, ERP, and MRP, it has become evident that OMS system is a different information system, and there should be done more research on this theme.

As a result, this work has focused on the design and development of a custom desktop application tailored specifically for a small car wash business. The application integrates a cash register and a management console. The analysis of existing software solutions and competing businesses in Narva further confirmed the necessity of a new system that aligns with requirements given by the future application's customer.

The development process also involved defining the core functionalities required by the end users, selecting suitable development tools, and implementing a user interface.

An important achievement of this project was the development of a prototype OMS kassa module, which was defined and implemented according to the specific needs of the business.

Ultimately, the work proves the importance of flexible and scalable software systems, especially for businesses.

KOKKUVÕTE

Kokkuvõtteks võib öelda, et käesolev lõputöö käsitles väikeste autopesuettevõtete ees seisvaid väljakutseid sobivate tarkvaralahenduste leidmisel, mis vastaksid nende konkreetsetele töövajadustele. Uuring näitas, et iga autopesula tegutseb oma unikaalse ärimudeli alusel, mistõttu ei saa "üks lahendus kõigile" lähenemist kõigi ettevõtete puhul rakendada. Erinevate infosüsteemide, nagu OMS, CRM, ERP ja MRP, analüüsi käigus ilmnes, et OMS on omaette infosüsteem ning selle teema uurimist tuleks kindlasti süvendada.

Selle töö põhitähelepanu oli suunatud spetsiaalselt väikese autopesuettevõtte jaoks kohandatud töölauarakenduse disainile ja arendamisele. Rakendus ühendab endas kassasüsteemi ja halduskonsooli. Olemasolevate tarkvaralahenduste ja Narvas tegutsevate konkureerivate ettevõtete analüüs kinnitas veelgi vajadust uue süsteemi järele, mis vastaks tulevase kliendi poolt seatud nõuetele.

Arendusprotsessi käigus määratleti lõppkasutajate jaoks vajalikud põhifunktsioonid, valiti sobivad arendustööriistad ning töötati välja kasutajaliides.

Üheks olulisemaks saavutuseks selles projektis oli OMS (Order Management System) kassamooduli prototüübi loomine, mis määratleti ja rakendati vastavalt konkreetse ettevõtte vajadustele.

Kokkuvõttes tõestab see töö paindlike ja skaleeritavate tarkvarasüsteemide olulisust, eriti just selliste ettevõtete puhul, kellel on spetsiifilised töövood ja vajadused.

REFERENCES

Harold, L & Whisler, T. (1958, November-December). Management in the 1980s. *Harvard Business Review* 36(6), p41-49

Gordon B. Davis. (1985). *Management Information Systems: Conceptual Foundations, Structure, and Development*. McGraw-Hill, Inc.

Gibson C. F. & Nolan R. L. Managing the Four Stages of EDP Growth (1974, January–February). *Harvard Business Review,* 52, p76-86

Nunamaker J. F. & Konsynski B. R. (1983). A DSS for Systems Analysis and Design. *Data Base Management: Theory and Applications*, p251-271, Reidel Publishing Company.)

Stair R., Reynolds G., Chesney T. (2008) Principles of Information Systems

Berry L. L., Shostack G.L., Upah G. D. (1983) *Emerging Perspectives on Services Marketing*, American Marketing Association

John Deighton. Siebel's Systems: Anatomy of Sales, Part 1 (2003, January 24) Harvard Business School

Al-Homery H.A., Ashari H., Ahmad A. (2023). Customer Relationship Management: A Literature Review Approach. *International Journal of Global Optimization and Its Application*, 2(1), p20-38.

Idzikowski A., Kuryło P., Cyganiuk J., (2019) Customer Relationship Management (CRM) - Philosophy and Its Significance for the Enterprise. *CzOTO*, 1(1), p1004-1011

Klochkova A. V., & Bebyakina A. A. (2019) CRM-система как инновационный инструмент повышения конкурентоспособности организации. *Научный журнал НИУ ИТМО. Серия Экономика и экологический менеджмент* 4, p177-184

Forrester J. W. (1965) *Industrial Dynamics*. M.I.T. PRESS

Olhager J (2013) Evolution of operations planning and control: from production to supply chains. *International Journal of Production Research*, 51(23-24), p6836-6843.

Klaus, Helmut, Rosemann, Michael, & Gable, Guy G. (2000) What is ERP? Information Systems Frontiers, 2(2), p141-162.

Chauhan V. & Singh J. (2017, June) Enterprise Resource Planning Systems for Service Performance in Tourism and Hospitality Industry. *International Journal of Hospitality & Tourism Systems*. 10(1), p57-62

Beer M., Spector B., Lawrence P. R., Mill D. Q., Walton R. E. (1984) *Managing Human Assets*. The Free Press

Armstrong M. (2006) *A Handbook of Human Resource Management Practice*. Kogan Page

Lambert D. M. (2004) *Supply Chain Management: Processes, Partnerships, Performance*. Supply Chain Management Institute

Lambert. D. M., Cooper M. C., Pagh J. D. (1998) Supply Chain Management: Implementation Issues and Research Opportunities. *The International Journal of Logistics Management*. 9(2), p1-19

Mentzer J. T., DeWitt W., Keebler J. S., Min S., Nix N. W., Smith C. D., Zacharia Z. G. (2001). Defining Supply Chain Management. *Journal of Business Logistics.* 22(2) p1-25.

A guide to order management systems (09.04.2025) https://www.microsoft.com/en-us/dynamics-365/topics/intelligent-order-management/order-management-system-oms

Order Management Systems Defined: What Is an OMS? (09.04.2025) https://www.netsuite.com/portal/resource/articles/erp/what-is-oms.shtml

What is order management? (09.04.2025) https://www.ibm.com/think/topics/order-management

Order Management System (OMS): Guide to Supply Chain Optimization For E-Commerce (09.04.2025) https://learn.subkit.com/order-management-system-oms-guide-to-supply-chain-optimization-for-e-commerce

Figas N. (09.04.2025) What Is an Order Management System (OMS) and How Does It Work? https://clearcode.cc/blog/what-is-order-management-system/

Order Management System (09.04.2025) https://onemoneyway.com/en/dictionary/order-management-system/