TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Lita Kornilova 195076IVSB

Building a Secure Mobile Application for Läänemaa County

Bachelor's thesis

Supervisor: Priidu Paomets

MSc

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Lita Kornilova 195076IVSB

Turvalise mobiilirakenduse loomine Läänemaale

Bakalaureusetöö

Juhendaja: Priidu Paomets
MSc

Tallinn 2022

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: Lita Kornilova

03.01.2022

Abstract

The purpose of this bachelor's thesis is to create a mobile application for Läänemaa county that would improve the information flow between the local government and residents. The application must be available for Android and iOS platforms and implement OWASP MASVS security standard.

The mobile application was implemented using Flutter cross-platform framework and allows the user to get an overview of events, dining places, news, services, transportation, and local government contacts.

A security audit for compliance with OWASP MASVS has been conducted and its outcome has been analysed.

As a result of the thesis, a working mobile application compliant with OWASP MASVS security standard has been developed and released to Google Play and App Store.

The thesis is written in English and contains 46 pages of text, 5 chapters, 12 figures and 10 tables.

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua mobiilirakendus, mis parandaks infoliikumist Läänemaa omavalitsuse ja selle elanike vahel. Rakendus peab olema kättesaadav nii Android kui ka iOS platvormidel ning vastama OWASP MASVS turvastandardile.

Mobiilirakendus on realiseeritud Flutteri platvormiülese raamistiku abil ning võimaldab selle kasutajatel saada ülevaade Läänemaa sündmustest, söögikohtadest, uudistest, teenustest, ühistranspordist ja kohalike omavalitsuste kontaktidest.

Läbi on viidud turvaaudit kontrollimaks rakenduse vastavust OWASP MASVS standarditele ja selle tulemuste analüüs.

Lõputöö tulemusena on välja töötatud OWASP MASVS turvastandarditele vastav töötav mobiilirakendus, mis on avalikult kättesaadav Google Play ja App Store keskkondades.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 46 leheküljel, 5 peatükki, 12 joonist, 10 tabelit.

List of abbreviations and terms

CMS	Content Management System
BLoC	Business Logic Component
SMTP	Simple Mail Transfer Protocol
UI	User interface
REST	Representational State Transfer
API	Application Programming Interface
OWASP	Open Web Application Security Project
MASVS	Mobile Application Security Verification Standard
HTTPS	Hypertext Transfer Protocol Secure
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

Table of contents

List of figures

List of tables

1 Introduction

1.1 Problem and background

Development of Läänemaa mobile application was ordered by Läänemaa county as a software tender where several companies had to make their offers. Based on the design wireframes and initial estimation, the company at which the author is employed [1] in partnership with an Estonian design company Rethink [2], have won the tender. The author of the thesis was assigned the role of mobile and backend developer. The author was the only software developer involved in the project.

The requirements for the mobile application were to provide users with the information concerning events happening in the county, available dining places, local services, and local government contacts. Users must also be able to read news, receive notifications when new articles are published and contact the local government. The mobile application must be developed for both Android and iOS platforms and be available in Estonian language.

The data was expected to come from various sources including two web applications and Läänemaa application's backend, however, this thesis does not cover development and security analysis of these components.

The client has emphasized that the developed application had to follow modern mobile software security practices. OWASP MASVS became the security standard of their choice due to its solid reputation and popularity. The applicable security level had to be assigned by the author based on the application's functionality.

Since the client has imposed the security requirements, the development team has decided to test whether the company's development practices comply with the OWASP MASVS basic security level (see section 4.1.1). To do that, it was decided to first develop the application not addressing the security standard and afterwards conduct a security audit on the finished software to identify whether the requirements have been met. All the

discovered non-compliant requirements had to be fixed before handing the application to the client.

1.2 Goals

The goal of the work is to develop and release Android and iOS applications that meets the functional requirement defined in the section 1.2.1 and implements the OWASP MASVS security standard.

1.2.1 Functional requirements

1.2.1.1 Events

User must be able to browse through the list of events which are fetched from midateha.visithaapsalu.com [3] (see Figure 3). User must be able to perform the following actions on event items:

- View event details, such as the event dates, description, and location
- Navigate to event homepage
- Filter events by date, category, and location

1.2.1.2 Dining places

User must be able to browse through the list of dining places, which are fetched from visithaapsalu.com [4] (see Figure 4). User must be able to perform the following action on dining places items:

- View dining place details, including location, preview image, category
- Navigate to the dining place homepage
- Filter dining places by category and location

1.2.1.3 News

User must be able to browse through the list of news which are inserted into the application's remote database from news portals of Läänemaa, Haapsalu, Vormsi and Lääne-Nigula (see Figure 5). User must be able to perform the following actions on news items:

- View news article styled text and details such as publication date, article's topic and affected region

- Filter news by topic and affected region
- See important alerts in a highlighted form on the application's dashboard
- Receive notifications about newly published news articles
- Subscribe to and unsubscribe from news from specific regions

1.2.1.4 Services

User must be able to browse through the list of services, which are inserted into the application's remote database (see Figure 6). User must be able to perform the following actions on service items:

- View service details, including contact details, location, description, opening times and category
- Navigate to the service homepage
- Filter services by category

1.2.1.5 Contacts

User must be able to browse through the local government contacts, which are inserted into the application's remote database (see Figure 7). User must be able to perform the following actions on contact items:

- View contact details including name, contact details, working schedule if available
- Navigate to the contact homepage
- Filter village elders alphabetically

1.2.1.6 Government notice

User must be able to send a notice to the local government about potential issues or to provide feedback (see Figure 8). User must be able to present their contact details, description of the problem and attach an image.

1.2.1.7 Transportation

User must be able to browse public transport related information, published in a form of articles (see Figure 9).

1.2.1.8 Favourites

User must be able to mark items as favourite for easy reference (see Figure 10). The items that can be added to favourites include events, dining places, services, local government contacts and news articles.

1.3 Methodology

To achieve the goal, the author had to analyse the application's functionality and choose the most appropriate technology stack considering that the application had to be developed for both Android and iOS platforms with the lowest price possible to meet the client's budget expectations.

After that, the author had to develop the application using the chosen technology stack. The application had to be developed with the use of reliable and reputed frameworks to avoid maintenance issues, the use of third-party libraries was allowed.

To meet the security requirements, the author has identified the applicable security level and conducted a security audit for compliance with OWASP MASVS standard. After the security issues have been revealed, the author has addressed them.

1.4 Thesis overview

The second part of the thesis introduces available mobile development technologies and explains the final technical stack choice.

The third part of the thesis describes the application development phase, provides the overview of its architecture and key components.

The fourth part of the thesis covers the application's security audit, evaluates its results, and suggests future improvements.

The fifth part of the thesis summarizes the outcome of the work.

2 Technologies

2.1 Mobile development approaches

Commonly there are identified 3 mobile development approaches – native, hybrid and cross-platform. Each of the approaches has its own benefits and disadvantages, which the author has presented in the Table 1 [5], [6], [7].

Table 1. Comparison of mobile development approaches

	Native	**Hybrid**	**Cross-platform**
Performance	Robust and efficient. The code is executed natively	Low performance compared to native, as the code is executed inside a WebView	Depending on the framework, performance can be comparable to native or lower
Hardware feature access	Easy and efficient. Available out of the box, no abstraction needed.	Hard and less efficient. Requires use of plugins or a native platform-specific implementation	Can use all the APIs exposed to native, however, new features and bug fixes might be available with a delay, since they will require the framework maintainer's attention
Debugging and profiling tools	Excellent tools which allow in-depth analysis, including CPU, memory, network, and battery usage profiling	Only web-specific tools are available, which might make platform specific issue mitigation more challenging	Solid toolkit, although native tools might be required for more in-depth analysis of platform-specific issues
Community and documentation	Excellent community and platform creator's support; excellent documentation; best practices directly from platform's creator	Excellent documentation and community for web-specific aspects, however the integration between web and native is much less supported	Community is smaller than for native, especially for newer frameworks; issues take longer to get resolved; some issues might need to be solved on the native level

	Native	Hybrid	Cross-platform
Development cost	Native developers are expensive. In addition, there are needed separate developers for each platform	Low cost, since most of the web code base can be reused in a mobile application, however, sometimes there might be a need for a native feature or plugin development	Depending on the framework, cross-platform developers may be slightly more or less expensive than native, since they are required to have knowledge both about the framework as well as about OS specific nuances
Development time	A separate application needs to be created for every platform. However, development speed per platform is quite high, since native provides easy access to platform-specific functionalities	Most of the code works out of the box on all platforms; web development frameworks are well established and there are well-known best practices, hence development time is low	Most of the code works across the platforms, however development time is higher compared to developing a native application for a single platform
Build size	Smallest out of all the options	Smaller or same as native. Since the application usually includes only a WebView and small amount of custom code, the download size is low	Large with applications developed using Flutter being the largest. Download size can be twice the size of a native application
Look and feel	Best possible. Native provides developers with a rich selection of components; there is a number of third-party libraries available as well. Application is responsive, graphic rendering is fast	There are many tools available for building web UI, however since the application is located inside the WebView, navigation between views might be clunky.	Depending on the framework, can be almost same as native or worse.

According to the comparison presented in the table, the following high-level conclusions can be made:

Native applications excel in performance, look, and feel, as well as reliability. This approach is best for complex feature rich applications. However, development cost is significantly higher compared to hybrid and cross-platform solutions.

Hybrid is the cheapest and fastest approach to implement, however the user experience might be lower compared to native solutions, as well as hybrid applications are limited in available features.

Cross-platform applications are a middle ground between native and hybrid applications with their performance and look being comparable to native and development cost being significantly lower.

2.2 Cross-platform mobile frameworks

There are several cross-platform mobile frameworks on the market with the most popular and trusted being Flutter, React native and Xamarin. A brief overview of them is presented in the Table 2.

Table 2. Overview of cross-platform mobile frameworks.

	React native	**Xamarin**	**Flutter**
Initial release	March 26, 2015	May 28, 2014	May 2017
Questions on Stack Overflow	118,791	47,231	107,008
Programming language	JavaScript	C#	Dart
Vendor	Facebook	Microsoft	Google
UI rendering	Different look throughout platforms	Can look identical or different throughout platforms	Can look identical or different throughout platforms

According to the table, Flutter is the newest framework, however the number of questions on Stack Overflow is comparable to React Native which was released two years earlier – that displays the active community of Flutter.

Another important factor is the way the frameworks render the UI. According to the design specifications, the UI must look the same way throughout the platforms, as well as it is relatively complex and contains a high number of non-standard components.

Figure 1 shows the beforementioned technologies' popularity over timer according to Google trends.

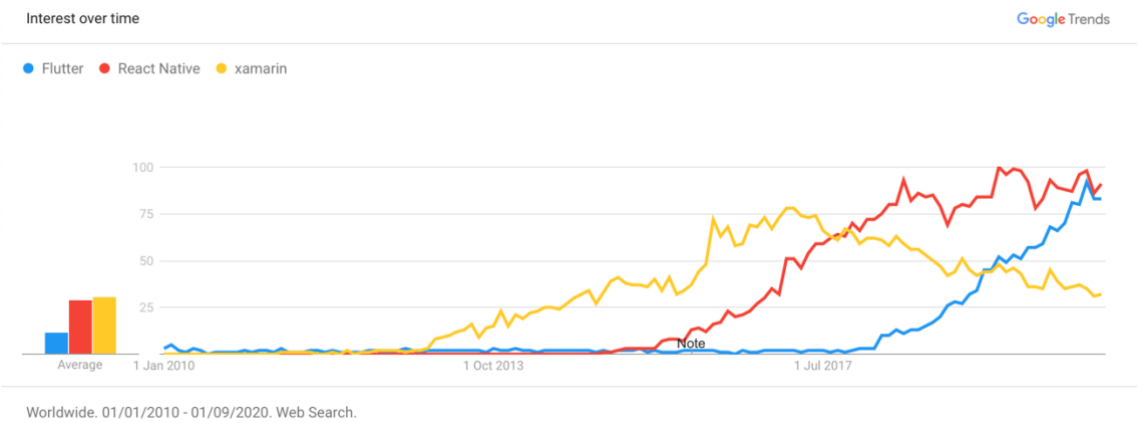


Figure 1. Cross-platform mobile development framework popularity according to Google Trends

Based on the presented chart, it is possible to make a conclusion that by the year 2020 when the Läänemaa project was started, only two of the analysed platforms popularity was not decreasing – React Native and Flutter. Demand for Xamarin's was steadily decreasing, as well as the developer community was relatively small.

Based on the popularity and community size, Xamarin was eliminated from candidate technologies, leaving the choice between Flutter and React Native.

A major difference between React Native and Flutter lies in the approaches these platforms utilize to draw user interfaces. React Native inherits native look of components, which means that a layout designed with React Native will look differently on Android and iOS devices, adapting to the platform. Flutter, on the other hand, draws identical user interface on both platforms.

The author has decided in favour of Flutter because the UI presented by the design team pictured the application equal on both platforms, which is more straightforward to achieve with Flutter.

3 Implementation

3.1 High level architecture

Although the thesis focuses only on the mobile application, for a better picture it is important to present the full architecture of the supporting services (see Figure 2). Each of the 11 components is briefly described below.

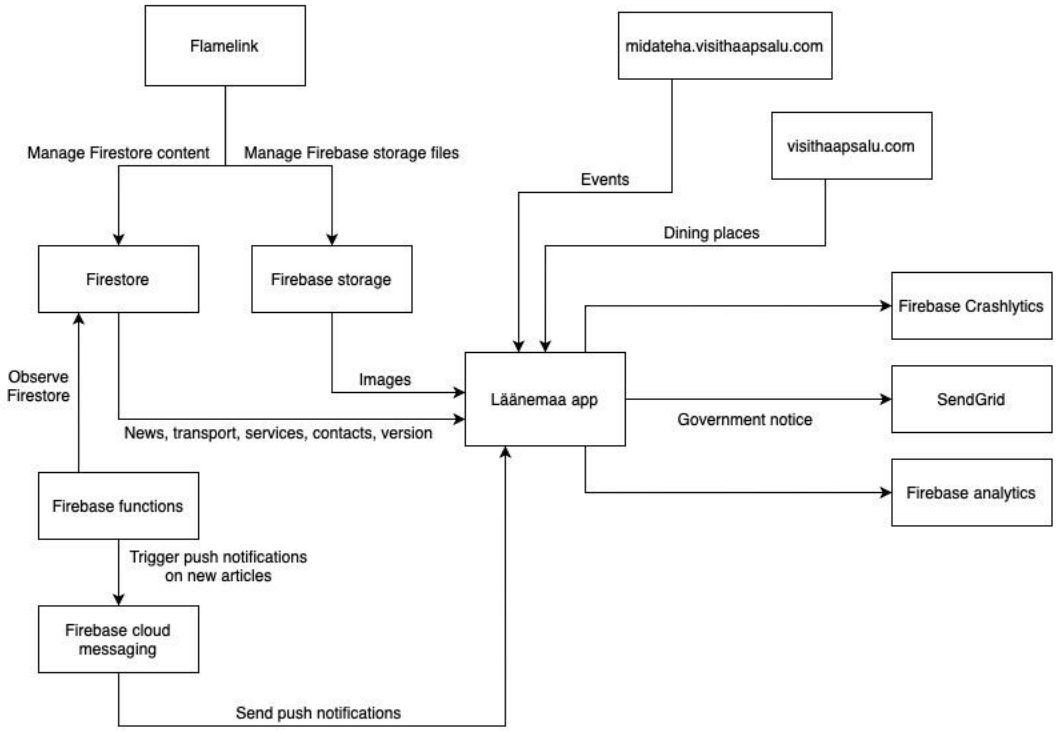


Figure 2. High level architecture of “Läänemaa” application infrastructure

3.1.1 “Läänemaa” application

“Läänemaa” is a cross platform mobile application built with Flutter and distributed for Android and iOS mobile devices. The application fetches data from midateha.visithaapsalu.com, visithaapsalu.com, Firebase Firestore and Firebase Storage.

3.1.2 Midateha.visithaapsalu.com

Midateha.visithaapsalu.com [3] is a WordPress-based website which provides users with information about upcoming events in Läänemaa county. The mobile application

communicates with it via a REST API which exposes the information available on the website. All the information is public and is accessed via GET requests.

3.1.3 Visithaapsalu.com

Visithaapsalu.com [4] is another WordPress-based website which provides users with various information concerning Läänemaa county, such as news, tourist attractions, and dining places. The mobile application retrieves list of dining places and associated metadata via a GET request through REST API. All the fetched data is publicly available.

3.1.4 Firebase services

Google Firebase is a Google-backed application development software platform that enables developers to use services that help to build iOS, Android, and web apps [8].

Firebase-based services play a significant role in the architecture of the application. They serve as an alternative to more conventional backend solutions. Firebase is designed to be used by application developers with no in-depth knowledge about infrastructure setup as opposed to, for example, Amazon Web Services (AWS) [9].

3.1.4.1 Firestore

Firestore [10] is a NoSQL document database in cloud that allows to query data directly from the application's code without an additional middle layer. Firebase serves as the primary database for the mobile application. The data stored in Firestore includes news articles, transportation, services, and contacts. The mobile application retrieves the data from Firestore using a Flutter Firestore client library (see the section 3.2.4).

3.1.4.2 Firebase storage

Firebase Storage [11] is a Firebase service which enables to store files in cloud. The files can be added, modified, and queried directly from the application code. "Läänemaa" application utilizes the Firebase storage benefits for storing images associated with news articles. The images are retrieved in the mobile application with the help of a Flutter client library (see the section 3.2.4)

3.1.4.3 Cloud functions for Firebase

Cloud functions for Firebase [12] is a service that allows to run backend code in response to custom defined actions, such as Firestore database transactions. In case of "Läänemaa" mobile application, Firebase functions are used to observe Firestore collection of news

articles and respond to the changes by triggering push notifications for Android and iOS applications.

3.1.4.4 Firebase Cloud Messaging

Firebase Cloud Messaging [13] is a cross-platform messaging solution which allows to send notification or data messages to and from an application. In “Läänemaa” app this service is used for sending users notifications when cloud functions trigger the cloud messaging client. Users have an option to fine-tune the notifications by subscribing to the regions of their interest (UI can be observed on the Figure 11).

3.1.4.5 Firebase Crashlytics

Firebase Crashlytics [14] is a crash reporting service which helps to track and analyse crash related issues in mobile applications. Both Android and iOS related crashes are reported to Crashlytics and are displayed in the Firebase Console under the Crashlytics tab. “Läänemaa” application uses Firebase Crashlytics to leverage timely response to potential bugs.

3.1.4.6 Firebase Analytics

Firebase Analytics [15] is a service which enables to report predefined and custom app events, such as number of daily active users, average engagement time, etc. In “Läänemaa” application, the analytics is used to record the default events as well as the user’s residence region: Lääne-Nigula, Vormsi, Haapsalu or unspecified. The selection of the region is done during onboarding, and it is not mandatory (UI can be observed on the Figure 12).

3.1.5 Flamelink

Flamelink [16] is a content management solution which is built to work in pair with Firebase Firestore. Flamelink provides a significantly more user-friendly interface for content managers, compared to editing content directly in Firebase Console. On the other hand, it eliminates the need to build a dedicated content management system with appropriate security measures.

In “Läänemaa” mobile application, Flamelink plays the role of a CMS which defines schemas for the following entities: news articles, services, contacts, transportation

articles, application version metadata. Content editors can as well use Flamelink to upload news article images to Firebase Storage.

3.1.6 Twilio SendGrid

Twilio SendGrid [17] is a customer communication platform which provides various email related features. In “Läänemaa” app, SendGrid is used to send email notifications to local government when a user submits a notice (UI can be observed on Figure 8). To avoid the overhead of adding a new library, SendGrid API is accessed from the mobile application through REST API.

3.2 Third-party libraries

This section is going to describe the third-party libraries used in the mobile application’s codebase that the author believes to play an important role in the overall architecture of the application. Other third-party utility libraries are purposefully omitted.

3.2.1 BLoC and Flutter BLoC

BLoC (Business Logic Components) [18] is a design pattern that helps to separate presentation from business logic using reactive approach of state management. BLoC logically divides the application into three layers:

- Presentation
- Business Logic
- Data

Each of the layers performs its own function and can be individually tested.

In Flutter, BLoC is presented in two libraries: bloc [19] that helps with the core pattern implementation, and flutter_bloc [20] that connects it with widgets for smooth integration.

The author has picked this design pattern as it is the recommended approach by Google, it is straightforward and helps to reduce complexity.

3.2.2 Retrofit.dart

Retrofit.dart [21] is a Dart HTTP client built on top of the low-level HTTP client Dio[22]. Retrofit.dart was inspired by the popular Android and Java library Retrofit [23]. It was chosen by the author due to the author's previous experience using Android Retrofit as well as its high popularity among Dart developers and frequent releases. The library utilizes automatic source code generation to minimize the amount of boilerplate code.

3.2.3 Floor

Floor [24] is a lightweight abstraction layer over SQLite for Flutter. Alike Retrofit.dart, it was inspired by an Android counterpart – Room [25] and was chosen by the author of the thesis for the same reasons as Retrofit.dart – frequent releases, high popularity, and previous experience with a similar technology.

3.2.4 FlutterFire libraries

FlutterFire [26] is a set of Flutter libraries released by Google's Firebase team that enable Flutter developers to easily use Firebase products without a need to write custom bridges for Android and iOS platforms. FlutterFire collection presents 14 stable Flutter plugins with different levels of support for mobile, web and desktop. "Läänemaa" application makes use of the following FlutterFire libraries:

- Firebase Core [27]
- Cloud Firestore [28]
- Firebase Storage [29]
- Firebase Messaging [30]
- Firebase Analytics [31]
- Firebase Crashlytics [32]

4 Security audit

4.1 Overview of OWASP MASVS

The Open Web Application Security Project (OWASP) is a non-profit foundation that works to improve the security of software [33]. The foundation is a collaborate effort of tens of thousands of security experts from all over the world. OWASP has been founded on December 1st, 2001, and nowadays is considered one of the most famous and trusted security communities. It defines more than 100 of active cyber security projects including standards, tool, documentation, code, and other type of projects [34], and any member is enabled to submit new proposals.

The OWASP Mobile Application Security Verification Standard (MASVS) [35] is a standard for mobile application security listed as a part of OWASP Mobile Security Testing Guide [36]. The standard is intended to be used by mobile software architects and developers as well as testers to ensure mobile application security at any stage of the product.

4.1.1 Verification types

OWASP MASVS defines two levels of application security: MASVS-L1 and MASVS-L2, where MASVS-L1 presents security requirements applicable to all mobile applications, whereas MASVS-L2 focuses on the applications that handle sensitive data or functionalities.

In addition to those, MASVS provides an opportunity to validate a mobile application for resiliency against reverse engineering and tampering. The resiliency requirements are grouped under MASVS-R.

One can combine security levels with resiliency requirements, which results in the verification types presented in the Table 3.

Table 3. Comparison of OWASP MASVS verification types

Verification type	Description
MASVS-L1	Basic security requirements applicable to all mobile applications
MASVS-L2	In-depth security requirements applicable to mobile applications handling sensitive data or functionalities, such as health care or financial software
MASVS L1+R	Basic security requirements applicable to all mobile applications with additional protection against client-side attacks. This verification type is often used for intellectual property protection or in gaming industry where it is seen essential to prevent cheating
MASVS L2+R	In-depth security requirements applicable to mobile applications handling sensitive data or functionality, that also require protection against client-side attacks. Applicable to the applications requiring state of the art security as well as resilience, such as financial industry applications and applications that actively use device storage and payment options, as opposed to delegating them to the server

4.1.2 Structure

OWASP MASVS is grouped into 8 chapters based on objective and scope. Each chapter consists of a list of security requirements where the first 7 chapters focus on Level 1 and Level 2 and the last one – on resiliency. Chapters include relevant definitions and references to testing guides.

4.2 Assigning the verification type

The first step in conducting a security audit is to assign the verification type relevant to the audited application. To do so the author must identify the desired security level (MASVS-L1 opposed to MASVS-L2) and whether the resiliency requirement is applicable.

Since the application does not handle any sensitive data or functionalities, it is reasonable to stick with MASVS-L1. As per resilience, the application does not hold any valuable intellectual property, as well as all the data accessed from the application, is publicly available, therefore the requirements for resiliency are not applicable.

Hence, the assigned verification type is MASVS-L1 which implies basic mobile application security requirements.

4.3 Validating security requirements

Below are presented only the requirements relevant to the MASVS-L1 verification type. The author is going to analyse the applicability of every requirement to the evaluated application as well as verify whether the applicable requirements are satisfied.

4.3.1 V1: Architecture, Design and Threat Modelling Requirements

The requirements specific to MASVS-L1 defined in the chapter V1: Architecture, Design and Threat Modelling Requirements are listed in the Table 4.

Table 4. OWASP MASVS-L1 V1: Architecture, Design and Threat Modelling Requirements.

#	MSTG-ID	Description	Applicable	Passed
1.1	MSTG-ARCH-1	All app components are identified and known to be needed.	yes	yes
1.2	MSTG-ARCH-2	Security controls are never enforced only on the client side, but on the respective remote endpoints.	yes	yes
1.3	MSTG-ARCH-3	A high-level architecture for the mobile app and all connected remote services has been defined and security has been addressed in that architecture.	yes	yes
1.4	MSTG-ARCH-4	Data considered sensitive in the context of the mobile app is clearly identified.	yes	yes
1.12	MSTG-ARCH-12	The app should comply with privacy laws and regulations.	yes	yes

4.3.1.1 MSTG-ARCH-1 and MSTG-ARCH-3

Both app components and remote services have been identified and described in the section 3.1. Each of the components has its designated purpose, which leaves the requirements MSTG-ARCH-1 and MSTG-ARCH-3 satisfied.

4.3.1.2 MSTG-ARCH-2

The remote services accessed by the application include

- Two WordPress-based sites owned by the Läänemaa local government's institutions
- Twilio SendGrid REST service for sending emails
- Firestore accessed via Flutter Firestore official library

In all the mentioned cases, the security controls are applied to the endpoints utilized by the mobile application.

4.3.1.3 MSTG-ARCH-4

In the context of MASVS, the following data is considered sensitive [37]:

- Personally identifiable information (PII) that can be abused for identity theft.
- Highly sensitive data that would lead to reputational harm and/or financial costs if compromised.
- Any data that must be protected by law or for compliance reasons.

Based on the definition, "Läänemaa" application does not collect or interact with any sensitive data. Hence, the requirement MSTG-ARCH-4 is satisfied.

4.3.1.4 MSTG-ARCH-12

The application complies with Estonian privacy laws and regulations. The privacy policy of the application is accessible inside the application under the "More" menu [38].

4.3.2 V2: Data Storage and Privacy Requirements

The requirements specific to MASVS-L1 defined in the chapter V2: Data Storage and Privacy Requirements are listed in the Table 5.

Table 5. OWASP MASVS-L1 V2: Data Storage and Privacy Requirements.

#	MSTG-ID	Description	Applicable
2.1	MSTG-STORAGE-1	System credential storage facilities need to be used to store sensitive data, such as PII, user credentials or cryptographic keys.	no
2.2	MSTG-STORAGE-2	No sensitive data should be stored outside of the app container or system credential storage facilities.	no
2.3	MSTG-STORAGE-3	No sensitive data is written to application logs.	no
2.4	MSTG-STORAGE-4	No sensitive data is shared with third parties unless it is a necessary part of the architecture.	no
2.5	MSTG-STORAGE-5	The keyboard cache is disabled on text inputs that process sensitive data.	no
2.6	MSTG-STORAGE-6	No sensitive data is exposed via IPC mechanisms.	no

#	MSTG-ID	Description	Applicable
2.7	MSTG-STORAGE-7	No sensitive data, such as passwords or pins, is exposed through the user interface.	no

As identified in the section 4.3.1.3, no sensitive data is handled by “Läänemaa” application, hence none of the requirements from the chapter V2: Data Storage and Privacy Requirements for L1 are deemed applicable.

4.3.3 V3: Cryptography Requirements

The requirements specific to MASVS-L1 defined in the chapter V3: Cryptography Requirements are listed in the Table 6.

Table 6. OWASP MASVS-L1 V3: Cryptography Requirements.

#	MSTG-ID	Description	Applicable
3.1	MSTG-CRYPTO-1	The app does not rely on symmetric cryptography with hardcoded keys as a sole method of encryption.	no
3.2	MSTG-CRYPTO-2	The app uses proven implementations of cryptographic primitives.	no
3.3	MSTG-CRYPTO-3	The app uses cryptographic primitives that are appropriate for the particular use-case, configured with parameters that adhere to industry best practices.	no
3.4	MSTG-CRYPTO-4	The app does not use cryptographic protocols or algorithms that are widely considered deprecated for security purposes.	no
3.5	MSTG-CRYPTO-5	The app doesn't re-use the same cryptographic key for multiple purposes.	no
3.6	MSTG-CRYPTO-6	All random values are generated using a sufficiently secure random number generator.	no

“Läänemaa” application does not utilize cryptographic approaches, hence none of the requirements from the chapter V3: Cryptography Requirements for L1 are applicable.

4.3.4 V4: Authentication and Session Management Requirements

The requirements specific to MASVS-L1 defined in the chapter V4: Authentication and Session Management Requirements are listed in the Table 7.

Table 7. OWASP MASVS-L1 V4: Authentication and Session Management Requirements

#	MSTG-ID	Description	Applicable
4.1	MSTG-AUTH-1	If the app provides users access to a remote service, some form of authentication, such as username/password authentication, is performed at the remote endpoint.	no
4.2	MSTG-AUTH-2	If stateful session management is used, the remote endpoint uses randomly generated session identifiers to authenticate client requests without sending the user's credentials.	no
4.3	MSTG-AUTH-3	If stateless token-based authentication is used, the server provides a token that has been signed using a secure algorithm.	no
4.4	MSTG-AUTH-4	The remote endpoint terminates the existing session when the user logs out.	no
4.5	MSTG-AUTH-5	A password policy exists and is enforced at the remote endpoint.	no
4.6	MSTG-AUTH-6	The remote endpoint implements a mechanism to protect against the submission of credentials an excessive number of times.	no
4.7	MSTG-AUTH-7	Sessions are invalidated at the remote endpoint after a predefined period of inactivity and access tokens expire.	no
4.12	MSTG-AUTH-12	Authorization models should be defined and enforced at the remote endpoint.	no

“Läänemaa” application does not require users to be authorized to access the application features; the data accessed by the application is publicly available in read-only mode and is not secured by access tokens.

Hence, none of the requirements from the chapter V4: Authentication and Session Management Requirements for L1 are applicable to the analysed application.

4.3.5 V5: Network Communication Requirements

The requirements specific to MASVS-L1 defined in the chapter V5: Network Communication Requirements are listed in the Table 8.

Table 8. OWASP MASVS-L1 V5: Network Communication Requirements

#	MSTG-ID	Description	Applicable	Passed
5.1	MSTG-NETWORK-1	Data is encrypted on the network using TLS. The secure channel is used consistently throughout the app.	yes	yes
5.2	MSTG-NETWORK-2	The TLS settings are in line with current best practices, or as close as possible if the mobile operating system does not support the recommended standards.	yes	yes
5.3	MSTG-NETWORK-3	The app verifies the X.509 certificate of the remote endpoint when the secure channel is established. Only certificates signed by a trusted CA are accepted.	yes	yes

Since the application utilizes exclusively HTTPS protocol, the requirement MSTG-NETWORK-1 is considered passed.

As per MSTG-NETWORK-2, the minimum allowed TLS version is set to 1.2 as by default Flutter security context settings [39]. As of January 2021, when the project was finalized, it was not possible to modify the lowest TLS version from the client code [40].

No self-signed certificates are allowed in the app. Only default trusted root certificates built into the respective platform [41] are whitelisted. That marks MSTG-NETWORK-3 as passed.

4.3.6 V6: Platform Interaction Requirements

The requirements specific to MASVS-L1 defined in the chapter V6: Platform Interaction Requirements are listed in the Table 9.

Table 9. OWASP MASVS-L1 V6: Platform Interaction Requirements

#	MSTG-ID	Description	Applicable	Passed
6.1	MSTG-PLATFORM-1	The app only requests the minimum set of permissions necessary.	yes	yes
6.2	MSTG-PLATFORM-2	All inputs from external sources and the user are validated and if necessary sanitized. This includes data received via the UI, IPC mechanisms such as intents, custom URLs, and network sources.	yes	yes

#	MSTG-ID	Description	Applicable	Passed
6.3	MSTG-PLATFORM-3	The app does not export sensitive functionality via custom URL schemes unless these mechanisms are properly protected.	no	-
6.4	MSTG-PLATFORM-4	The app does not export sensitive functionality through IPC facilities, unless these mechanisms are properly protected.	no	-
6.5	MSTG-PLATFORM-5	JavaScript is disabled in WebView's unless explicitly required.	no	-
6.6	MSTG-PLATFORM-6	WebView's are configured to allow only the minimum set of protocol handlers required (ideally, only https is supported). Potentially dangerous handlers, such as file, tel and app-id, are disabled.	no	-
6.7	MSTG-PLATFORM-7	If native methods of the app are exposed to a WebView, verify that the WebView only renders JavaScript contained within the app package.	no	-
6.8	MSTG-PLATFORM-8	Object deserialization, if any, is implemented using safe serialization APIs.	yes	yes

4.3.6.1 MSTG-PLATFORM-1

The application requests different sets of permissions on Android and iOS due to the differences in permission handling between the platforms. On Android, two permissions are requested: the INTERNET permission is requested at install time and the WRITE_EXTERNAL_STORAGE permission is requested when user tries to perform an image uploading action. On iOS, access to library is requested at install time and push notifications permission is requested during onboarding. Therefore, the requirement is deemed as satisfied.

4.3.6.2 MSTG-PLATFORM-2

The application does not utilize intents, custom URLs, and external network sources. The only functionality in the application requiring the user input is a form for communication with local government – all the inputs in the form are validated to be not empty. No additional validation mechanisms are implemented to simplify the user flow. The input is

passed directly to the SMTP provider. Based on that, the requirement MSTG-PLATFORM-2 is evaluated as passed.

4.3.6.3 MSTG-PLATFORM-3 and MSTG-PLATFORM-4

As identified in the section 4.3.1.3, “Läänemaa” application does not contain any sensitive functionality. Hence, the requirements concerning sensitive functionalities exposure are not applicable.

4.3.6.4 MSTG-PLATFORM-5 to MSTG-PLATFORM-7

“Läänemaa” application contains a way to open web pages, however they are launched in external browser as opposed to WebView’s inside the app. Since the application does not make use of WebView’s, the requirements MSTG-PLATFORM-5 to MSTG-PLATFORM-7 do not apply.

4.3.6.5 MSTG-PLATFORM-8

The application takes advantage of the Google’s “json_serializable” serialization API [42]. The deserialization is implemented for the data coming from REST API - dining places, events, and Google Places - all of which are fetched from trusted sources. Deserialization of data coming from Firebase Storage is performed by the firebase storage Flutter client and is provided to the application in a shape of a map of strings to dynamic objects. Based on the information explained above, the requirement MSTG-PLATFORM-8 is considered as implemented.

4.3.7 V7: Code Quality and Build Setting Requirements

The requirements specific to MASVS-L1 defined in the chapter V7: Code Quality and Build Setting Requirements are listed in the Table 10.

Table 10. OWASP MASVS L1-V7: Code Quality and Build Setting Requirements

#	MSTG-ID	Description	Applicable	Passed
7.1	MSTG-CODE-1	The app is signed and provisioned with a valid certificate, of which the private key is properly protected.	yes	yes
7.2	MSTG-CODE-2	The app has been built in release mode, with settings appropriate for a release build (e.g., non-debuggable).	yes	yes

#	MSTG-ID	Description	Applicable	Passed
7.3	MSTG-CODE-3	Debugging symbols have been removed from native binaries.	yes	yes
7.4	MSTG-CODE-4	Debugging code and developer assistance code (e.g., test code, backdoors, hidden settings) have been removed. The app does not log verbose errors or debugging messages.	yes	yes
7.5	MSTG-CODE-5	All third-party components used by the mobile app, such as libraries and frameworks, are identified, and checked for known vulnerabilities.	yes	yes
7.6	MSTG-CODE-6	The app catches and handles possible exceptions.	yes	yes
7.7	MSTG-CODE-7	Error handling logic in security controls denies access by default.	yes	yes
7.8	MSTG-CODE-8	In unmanaged code, memory is allocated, freed, and used securely.	no	-
7.9	MSTG-CODE-9	Free security features offered by the toolchain, such as bytecode minification, stack protection, PIE support and automatic reference counting, are activated.	yes	yes

4.3.7.1 MSTG-CODE-1

The Android application is distributed as an application bundle [43] with using V1 and V2 signing schemes. The iOS application is signed using Apple issued certificates [44]. In both cases, the applications are signed and distributed using the platform recommended approaches, hence the requirement MSTG-CODE-1 is deemed as satisfied.

4.3.7.2 MSTG-CODE-2

Both the Android and iOS applications are built in release mode with the appropriate release settings, i.e., minification and proguard in case of Android, hence the requirement MSTG-CODE-2 is considered satisfied.

4.3.7.3 MSTG-CODE-3

On Android when building with Gradle and on iOS when assembling a distribution build, debug symbols are removed by default. Hence, the requirement MSTG-CODE-3 is satisfied.

4.3.7.4 MSTG-CODE-4

All the debugging code has been removed from the application before releasing. The logging is configured to only produce logs in debug builds. Therefore, the requirement MSTG-CODE-4 is passed.

4.3.7.5 MSTG-CODE-5

The third-party libraries utilized in the application are distributed by reputable developers and hold to security standards. The author of the thesis has verified the security of the libraries before making them part of the application code, for this reason the requirement MSTG-CODE-5 is considered satisfied.

4.3.7.6 MSTG-CODE-6

The exceptions that might occur during the network requests, database operations and unchecked mappings, are caught and handled. Therefore, the requirement MSTG-CODE-6 is considered satisfied.

4.3.7.7 MSTG-CODE-7

The error handling logs are handled by Firebase Crashlytics and are only available to the user authorized in Firebase console. Therefore, the requirement MSTG-CODE-7 is deemed as satisfied.

4.3.7.8 MSTG-CODE-8

The requirement MSTG-CODE-8 is not applicable in case of “Läänemaa” app, since the term of “unmanaged code” refers to C/C++ languages, and hence applies to Android NDK or Xamarin based applications, which is not the case of the analysed Flutter application.

4.3.7.9 MSTG-CODE-9

The default security features offered by the Flutter framework and Android and iOS building tools are enabled – therefore the requirement MSTG-CODE-9 is met.

4.4 Security audit results

As it was stated in the Problem and background, the development team agreed to use “Läänemaa” project as an opportunity to test internal development practices for compliance with basic security requirements.

During the security audit for compliance with MASVS-L1, the author evaluated 46 requirements, out of which 19 were deemed as applicable and 19 were marked passed.

According to the numbers, “Läänemaa” application met all the applicable security requirements, which shows that the company’s development practices provided a successful outcome.

No security patches were required.

Although the results of the test are highly satisfactory, that might not have been the case for a more feature rich application. An application handling user authentication and processing user entered, or sensitive data would require more attention to guidance of security standards. Developers need to be trained about the basic level security requirements and security audits must be conducted as part of standard testing process.

5 Summary

The purpose of the work was to develop a mobile application for Läänemaa county which would help to enhance the information flow between the local government and residents of the county. The application must be available for Android and iOS platforms and must implement OWASP MASVS mobile security standard.

The author studied available mobile development approaches, compared them, and selected cross-platform approach as the most suitable for the aim of the project. Flutter was chosen as the cross-platform framework for its popularity and the way it renders UI on different platforms.

As a result, the author has developed the application that meets the client's functional requirements and follows the UI specifications provided by the design team.

After the implementation phase, the author has conducted a security audit for compliance with OWASP MASVS security standard. As a result of the audit, it was concluded that the developed application implements the standard and the client approved it for release.

As a result, the goal of the work was achieved, and the application was successfully published on the App Store and Google Play Store (see Appendix 2 for information).

After closing the project, the client has contacted the company where the author is employed for minor fixes, as well as presented some ideas for new features. Next development phase is currently in the process of negotiation.

References

[1] FOB Solutions homepage. Available: https://www.fob-solutions.com. [Accessed 21 December 2021].

[2] Rethink homepage. Available: https://rethink.ee. [Accessed 21 December 2021].

[3] Events in Läänemaa homepage. Available: https://midateha.visithaapsalu.com. [Accessed 21 December 2021].

[4] Läänemaa tourism homepage. Available: https://www.visithaapsalu.com. [Accessed 21 December 2021].

[5] Rakesh Patel, "Different Types of Mobile Apps?", 2021. Available: https://www.spaceo.ca/types-of-mobile-apps/. [Accessed 16 December 2021].

[6] Clearbridge Mobile, "A Guide to Mobile App Development: Web vs. Native vs. Hybrid", 2020. Available: https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/. [Accessed 16 December 2021].

[7] Satinder Singh, "Native vs Hybrid vs Cross Platform – What to Choose in 2022?", 2021. Available: https://www.netsolutions.com/insights/native-vs-hybrid-vs-cross-platform/. [Accessed 16 December 2021].

[8] Linda Rosencrance, "Google Firebase", 2019. Available: https://searchmobilecomputing.techtarget.com/definition/Google-Firebase. [Accessed 26 December 2021].

[9] Taavi Rehemägi, "Firebase vs AWS – Is It Even a Fair Fight?", 2020. Available: https://dashbird.io/blog/aws-lambda-vs-firebase/. [Accessed 26 December 2021].

[10] Firebase Firestore. Available: https://firebase.google.com/products/firestore. [Accessed 26 December 2021].

[11] Firebase Cloud Storage. Available: https://firebase.google.com/products/storage. [Accessed 26 December 2021].

[12] Firebase Cloud Functions. Available: https://firebase.google.com/products/functions. [Accessed 26 December 2021].

[13] Firebase Cloud Messaging. Available: https://firebase.google.com/products/cloud-messaging. [Accessed 26 December 2021].

[14] Firebase Crashlytics. Available: https://firebase.google.com/products/crashlytics. [Accessed 26 December 2021].

[15] Firebase Analytics. Available: https://firebase.google.com/products/analytics. [Accessed 26 December 2021].

[16] Flamelink homepage. Available: https://flamelink.io. [Accessed 26 December 2021].

[17] Twilio SendGrid homepage. Available: https://www.twilio.com/sendgrid/email-api. [Accessed 26 December 2021].

[18] BLoC pattern homepage. Available: https://bloclibrary.dev/. [Accessed 26 December 2021].

[19] BLoC project homepage. Available: https://pub.dev/packages/bloc. [Accessed 26 December 2021].

[20] Flutter BLoC project homepage. Available: https://pub.dev/packages/flutter_bloc. [Accessed 26 December 2021].

[21] Flutter Retrofit project homepage. Available: https://pub.dev/packages/retrofit. [Accessed 28 December 2021].

[22] Dio project homepage. Available: https://pub.dev/packages/dio. [Accessed 28 December 2021].

[23] Retrofit project homepage. Available: https://square.github.io/retrofit/. [Accessed 28 December 2021].

[24] Floor project homepage. Available: https://pub.dev/packages/floor. [Accessed 28 December 2021].

[25] Room project homepage. Available: https://developer.android.com/jetpack/androidx/releases/room. [Accessed 28 December 2021].

[26] FlutterFire homepage. Available: https://firebase.flutter.dev. [Accessed 2 January 2022].

[27] Firebase Core Flutter plugin homepage. Available: https://pub.dev/packages/firebase_core. [Accessed 14 December 2021].

[28] Cloud Firestore Flutter plugin homepage. Available: https://pub.dev/packages/cloud_firestore. [Accessed 14 December 2021].

[29] Firebase Storage Flutter plugin homepage. Available: https://pub.dev/packages/firebase_storage. [Accessed 14 December 2021].

[30] Firebase Messaging Flutter plugin homepage. Available: https://pub.dev/packages/firebase_messaging. [Accessed 14 December 2021].

[31] Firebase Analytics Flutter plugin homepage. Available: https://pub.dev/packages/firebase_analytics. [Accessed 14 December 2021].

[32] Firebase Crashlytics Flutter plugin homepage. Available: https://pub.dev/packages/firebase_crashlytics. [Accessed 14 December 2021].

[33] Open Web Application Security Project, “About the OWASP Foundation”. Available: https://owasp.org/about/. [Accessed 26 December 2021].

[34] OWASP - Projects. Available: https://owasp.org/projects/. [Accessed 26 December 2021].

[35] OWASP MASVS Github page. Available: https://github.com/OWASP/owasp-masvs. [Accessed 26 December 2021].

[36] OWASP Mobile Security Testing Guide, “Mobile App Security Requirements and Verification”, 2021. [Online]. Available https://owasp.org/www-project-mobile-security-testing-guide/. [Accessed 26 December 2021].

[37] Open Web Application Security Project, “OWASP MASVS 1.3”, 2021. Available: https://github.com/OWASP/owasp-masvs/releases/download/v1.3/OWASP_MASVS-1.3-en.pdf. [Accessed 26 April 2021].

[38] Läänemaa privacy policy page. Available: https://laanemaa.ee/privaatsustingimused/. [Accessed 02 January 2021].

[39] Dart language source code, “Context security”. Available: https://github.com/dart-lang/sdk/blob/bbde6ba9b3c8ef4b86c5096d3b836c6490eca280/runtime/bin/security_context.cc#L810. [Accessed 26 December 2021].

[40] Dart SDK issues, “SecurityContext with minimal protocol version”. Available: https://github.com/dart-lang/sdk/issues/37173. [Accessed 26 December 2021].

[41] Flutter API documentation, “Security Context – defaultContext property”. Available: https://api.flutter.dev/flutter/dart-io/SecurityContext/defaultContext.html. [Accessed 24 December 2021].

[42] Json Serializable plugin homepage. Available: https://pub.dev/packages/json_serializable. [Accessed 24 December 2021].

[43] Android documentation – App Bundle. Available: https://developer.android.com/guide/app-bundle. [Accessed 24 December 2021].

[44] Apple documentation – Certificates. Available: https://developer.apple.com/support/certificates/. [Accessed 24 December 2021].

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

I Lita Kornilova

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis " Building a secure mobile application for Läänemaa county ", supervised by Priidu Paomets

1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;

1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.

3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

03.01.2022

Appendix 2 – Screenshots of “Läänemaa” mobile application

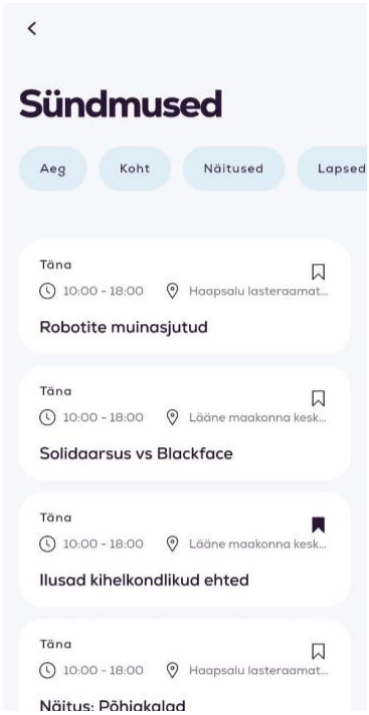


Figure 3. Events screen in “Läänemaa” application

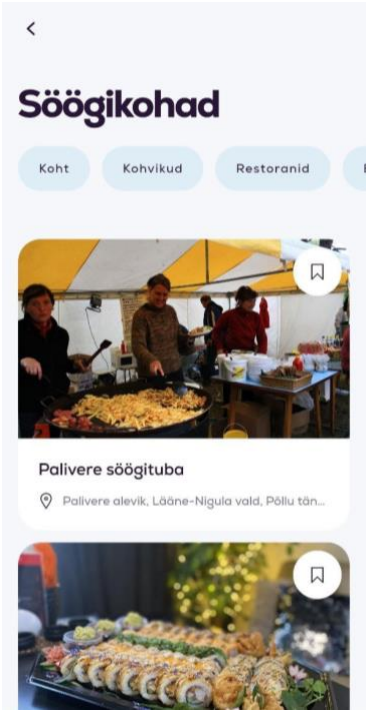


Figure 4. Dining places screen in “Läänemaa” application

Figure 5. News screen in “Läänemaa” application

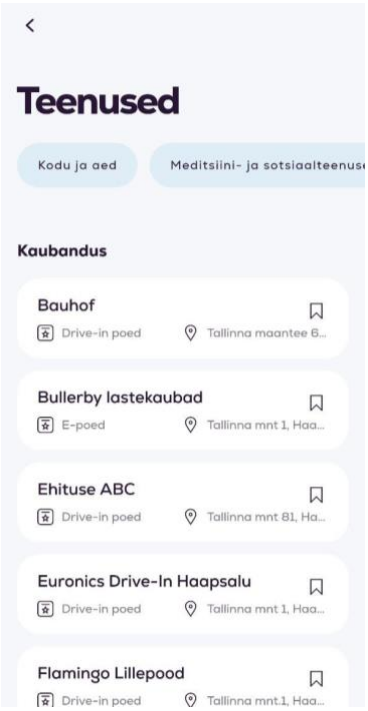


Figure 6. Services screen in “Läänemaa” application

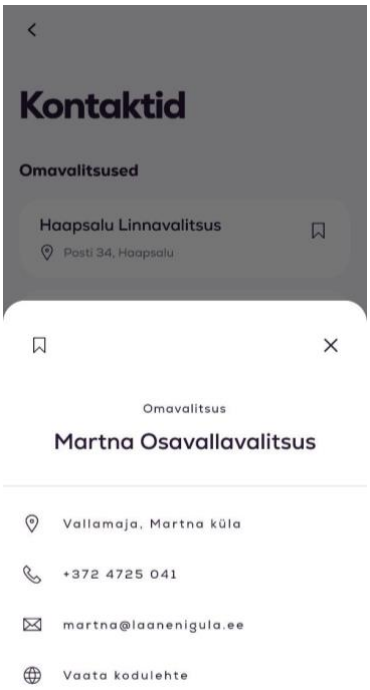


Figure 7. Contacts details view in “Läänemaa” application

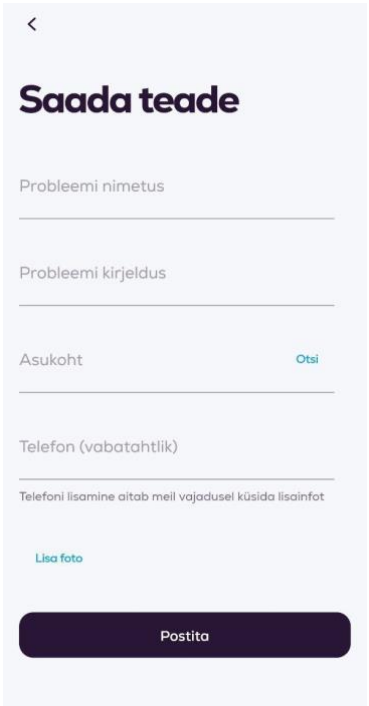
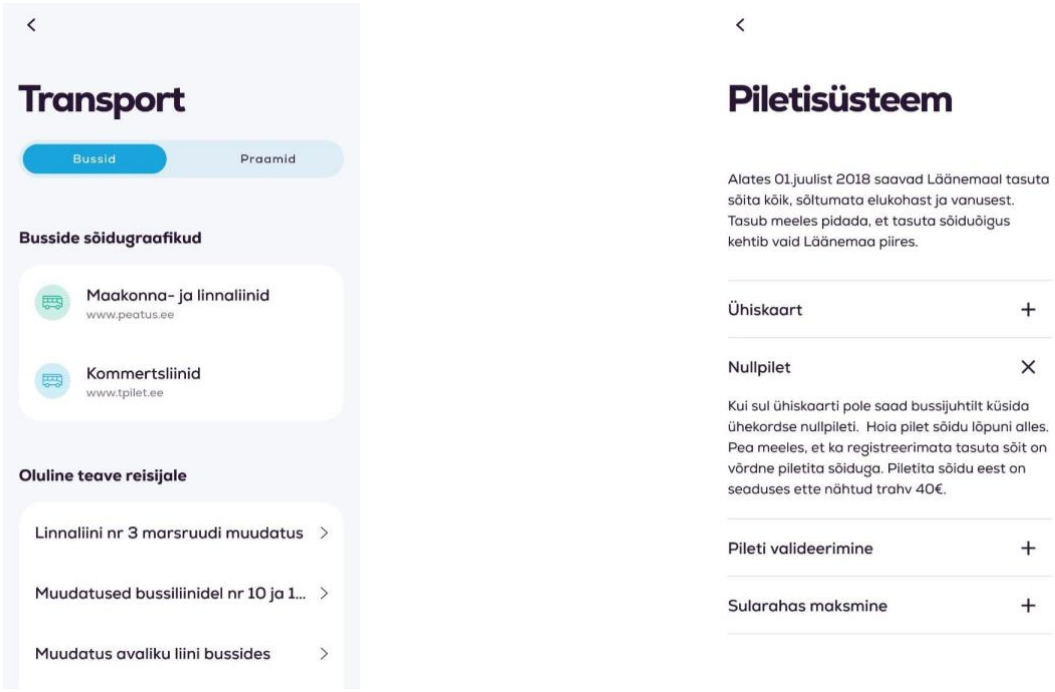


Figure 8. Notice form screen in “Läänemaa” application



(a) (b)

Figure 9. Transport screens: (a) list view, (b) content view in “Läänemaa” application

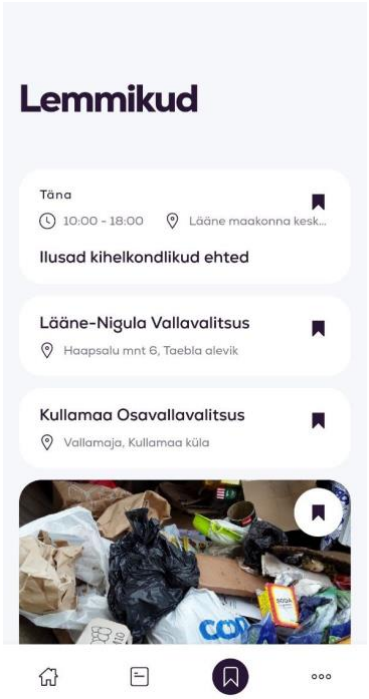


Figure 10. Favourites screen in “Läänemaa” application

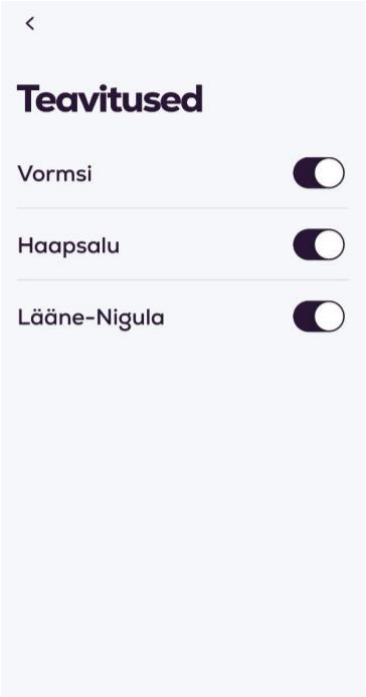


Figure 11. Notification settings screen in “Läänemaa” application

Figure 12. Onboarding region screen in “Läänemaa” application

Appendix 2 – Mobile application in Play Store and App Store

The mobile application was uploaded to the Google Play Store and iOS App store under the name “Läänemaa”, published by SA Läänemaa. It is available on the following URLs:

- https://apps.apple.com/sa/app/läänemaa/id1542797570
- https://play.google.com/store/apps/details?id=ee.laanemaa