BACHELOR'S THESIS 2021

En jämförande studie av cross-platform-ramverk

Rilind Zejnullahu, Mohammed Menim

ISSN 1651-2197
LU-CS/HBG-EX: 2021-04

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

En jämförande studie av cross-platform-ramverk

LUNDS UNIVERSITET
Campus Helsingborg

LTH Ingenjörshögskolan vid Campus Helsingborg
Institutionen för datavetenskap

Examensarbete:
Rilind Zejnullahu
Mohammed Menim

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Lunds universitet
Lund 2021

Sammanfattning

Idag finns det många olika ramverk som kan användas för att utveckla applikationer. Det finns både nativeapplikationer och cross-platform-applikationer. Nativeapplikationer utvecklas för ett specifikt operativsystem medan cross-platform-applikationer ska fungera för flera operativsystem. Vilket ramverk man borde använda kan vara svårt att bestämma för utvecklare.

Inom ramen för detta examensarbete jämförs två cross-platform-ramverk, Uno Platform och Flutter. Både är relativt nya och har både fördelar och nackdelar. Det utvecklades två prototyper med ramverken där syftet är att jämföra ramverken. Genom att utveckla prototyperna kunde examensarbetarna komma fram till en slutsats om vilket ramverk är bättre utifrån ett antal kriterier som valdes ut. Kriterierna som valdes var: programmeringsspråk, kod, storlek, kompileringstid, tid att hämta data från API, prestanda, hot reload, paket och tillgängligheten av information. Efter att under en månad ha utvärderat respektive ramverk blev slutsatsen att Flutter är ett bättre ramverk än Uno Platform utifrån kriterierna. Största skillnaderna var prestandan och tillgängligheten av information, där Flutter hade en stor fördel. Det var lättare att hitta information om Flutter än vad det var om Uno Platform. Vidare renderas komponenter i Flutter snabbare än komponenter i Uno Platform.

Examensarbetet gjordes i samarbete med Softhouse, vilket är ett företag som bland annat utvecklar applikationer, främst mobilapplikationer. Resultatet av detta examensarbete är intressant för dem eftersom dessa är nya ramverk som kan eventuellt ersätta ramverken som företaget använder idag.

Nyckelord: Flutter, Uno Platform, cross-platform-ramverk, nativeapplikationer

Abstract

Today there are a lot of different frameworks that can be used to develop applications. There are both native applications and cross-platform applications. Native applications are developed for a specific operating system while cross-platform applications should work for multiple operating systems. What framework should be used can be hard to decide for developers.

Within the framework of this thesis, two cross-platform frameworks are compared, Uno Platform and Flutter. Both are relatively new and they come with both pros and cons. Two prototypes were developed using the frameworks with the purpose of comparing the frameworks. By developing the prototypes, a conclusion could be drawn by the authors of this thesis of which framework is better according to a selection of criterias. The criterias that were chosen were programming language, code, size, compilation time, time to get data from API, performance, hot reload, packages and the availability of information. After a month of evaluating the respective frameworks, the conclusion was that Flutter is a better framework than Uno Platform according to the criterias. The biggest differences were the performance and the availability of information, where Flutter had a big advantage. It was easier to find information about Flutter than it was about Uno Platform. Also, components render faster in Flutter compared to the components in Uno Platform.

The thesis was done in collaboration with Softhouse, a company that among other things develops applications, mainly mobile applications. The result of this thesis is of interest to them because these are new frameworks that eventually can replace the frameworks that the company is currently using today.

Keywords: Flutter, Uno Platform, cross-platform frameworks, native application

Förord

Detta examensarbete utfördes i samarbete med Softhouse i Malmö. Examensarbetet har gett oss tillfället att komma ut och pröva på arbetslivet samt öka våra kunskaper inom programmering men också vår förståelse för dagens cross-platform-teknik.

Ett stort tack till vår handledare från Lunds tekniska högskola, Christian Nyberg, som har besvarat samtliga frågor som vi har ställt samt väglett oss under examensarbetets gång. Vidare vill vi även tacka vår handledare från Softhouse, Lars Isberg som har hjälpt och gett oss tips under utvecklingen av de två prototyperna samt chefen från Softhouse, Ola Persson som gett oss möjligheten att få göra examensarbetet med Softhouse. Sist men inte minst vill vi tacka Christin Lindholm som har varit vår examinator under examensarbetet. Det har varit ett stort nöje att ha Christian och Lars som våra handledare, Ola som anordnare för examensarbetet och Christin som examinator.

Innehållsförteckning

1 Inledning

I detta kapitlet får man en inledning över examensarbetet som har genomförts tillsammans med Softhouse, där målet är att jämföra ett eller flera cross-platform-ramverk med ramverket Uno Platform. Kapitlet tar upp bakgrund, syfte och målformulering. Vidare resoneras det kring problemformulering, motivering av examensarbetet samt avgränsningar.

1.1 Bakgrund

Detta examensarbete görs i samarbete med företaget Softhouse [1] som arbetar inom lean och agil mjukvaruutveckling. Företaget har flera kontor runt om i Sverige samt ett i Sarajevo. Detta examensarbetet utförs på Softhouse i Malmö. Då de utvecklar applikationer är resultatet av detta examensarbete intressant för dem.

Softhouse utvecklar både app-lösningar och webblösningar och vill utveckla dessa med bästa möjliga teknologier beroende på vilka krav som ställts. Några exempel på app-lösningar som Softhouse utvecklat är Skånetrafikens app, Go-Ahead som tillhör Go-Ahead Nordic samt en mobil biljettlösning för företaget Flytoget. Angående webblösningar har de utvecklat en för ett företag vid namnet IST. Denna webblösning är responsiv och är tillgänglig för smartphones, surfplattor och datorer. Därför är det intressant för företaget att undersöka och jämföra olika möjliga teknologier för att utveckla applikationer, speciellt då nya ramverk introduceras.

Idag finns många olika ramverk som kan utnyttjas för att utveckla applikationer. Vilket ramverk man ska använda beror på vilka mål man vill uppnå med applikationen. Det finns så kallade nativeapplikationer, vilket är en applikation som har utvecklats för ett specifikt operativsystem som till exempel Android eller iOS [2]. Det finns även möjligheten att utveckla applikationer med hjälp av cross-platform-ramverk. Dessa ramverk möjliggör att man kan utveckla en kodbas som sedan ska fungera i flera operativsystem [2].

Ett problem utvecklare möter är att det finns många ramverk att välja mellan. Här är bara några av de olika cross-platform-ramverk som används idag: Xamarin Forms, React Native, Flutter, Adobe PhoneGap, Ionic, Uno Platform med flera. Det kan vara svårt som utvecklare att bestämma sig för vilket ramverket som bör användas för att utveckla en applikation.

Uno Platform är ett av ramverken som ska undersökas eftersom att Softhouse var intresserade att undersöka ramverket. Med hjälp av Uno Platform kan man utveckla en applikation som fungerar för iOS, macOS och Android. Detta ramverk bygger på Xamarin Native, inte att förväxlas med ramverket Xamarin Forms som även det bygger på Xamarin Native. Med Uno Platform kan man även skapa en webbapplikation, den använder sig utav WebAssembly för att bygga webbapplikationer. Applikationen fungerar även på Linux där den använder Skia för att skapa själva användargränssnittet. Allt som krävs är en kodbas där koden är baserat på WinUI och Universal Windows Platform [3].

En del av examensarbetet är att utreda vilket eller vilka ramverk som Uno Platform ska jämföras med. Två möjliga alternativ är Flutter eller Ionic. Examensarbetet kommer inledas med en initial undersökning där det utreds vilket eller vilka ramverk som Uno Platform bör jämföras med, urvalet kommer att ske utifrån ett antal kriterier. Sedan kommer det göras en djupgående jämförelse mellan Uno Platform och ramverken eller ramverket som valdes ut. Den djupgående jämförelsen kommer att göras utifrån ett antal andra kriterier.

Problemen som detta examensarbete löser är att Softhouse får en bättre blick utav ramverket Uno Platform som Softhouse var intresserade att undersöka. Vidare får Softhouse en detaljerad jämförelse mellan Uno Platform och det andra ramverket. Jämförelsen kan i sin tur användas för att Softhouse ska kunna avgöra om någon av de två ramverken passar bra för företagets framtida utveckling av applikationer. Det som är nytt med examensarbetet är att det är relativt nya ramverk som undersöks.

1.2 Syfte

Softhouse kommer ha nytta av detta examensarbete då man kommer komma fram till en slutsats om någon av dessa cross-platform-ramverk är ramverk som företaget kan tänkas använda för att utveckla deras framtida applikationer. Syftet med detta examensarbete är att utvärdera två eller flera olika cross-platform-ramverk.

1.3 Målformulering

Slutmålet inom detta examensarbete är att dra en slutsats angående vilket ramverk som är bäst utifrån ett antal kriterier för att sedan kunna lämna en rekommendation till företaget. För att uppnå detta mål ska det utifrån ett antal kriterier väljas ut vilka ramverk som ska vara med i den djupgående jämförelsen

med Uno Platform. Det ska göras en djupgående jämförelse utifrån ett antal kriterier som väljs ut. Prototyper ska utvecklas som en del i jämförelsen av de olika ramverken.

1.4 Problemformulering

Nedan presenteras de frågeställningarna som kommer att besvaras under examensarbetets gång.

1. Vilka kriterier är lämpliga att ha för den initiala jämförelsen?
2. Vilka kriterier är lämpliga att ha vid den djupgående jämförelsen mellan ramverken som valdes ut i den initiala jämförelsen och Uno Platform?
3. Vilken typ av applikation är lämplig som prototyp? Vilken funktionalitet ska finnas med?

1.5 Motivering av examensarbetet

Företaget var intresserat av Uno Platform, vilket är relativt ny då den introducerades två år sedan (7 maj 2018). Efter att ha tagit reda på information om ramverket, var intresset att utvärdera ramverket ömsesidigt. För att detta skulle passa för ett examensarbete valdes det att göra en jämförelse med ett annat ramverk. Detta tycktes passa bra eftersom en jämförelse kan ge en bra bild av ramverkets potential. Dessutom tycktes det även vara intressant med apputveckling. Utifrån detta examensarbete kommer kunskaperna inom ett område som är intressant att utvidgas.

Man kan även argumentera för att resultatet av examensarbetet kan leda till bättre applikationer för samhället. Om man till exempel kommer fram till att Uno Platform är bättre än ett annat ramverk enligt vissa kriterier, kommer kanske Softhouse eller andra utvecklare välja att använda Uno Platform istället för ett annat ramverk.

1.6 Avgränsningar

När det kommer till Uno Platform ska det fokuseras mer på WebAssembly-delen, vilket innebär att man inte bygger något i native och det kommer därför inte finnas mobilspecifika funktioner. Detta görs då företaget redan har erfarenhet av Xamarin, vilket Uno Platform bygger på. De har även ett speciellt intresse för just cross-platform-ramverk som inte kräver att man bygger något i native. Utvecklingen av prototyperna kommer ske på datorer med Windows 10 och prototyperna kommer att köras på Google Chrome.

2 Teknisk bakgrund

Det här kapitlet beskriver den tekniska bakgrunden för detta examensarbete.

2.1 Cross-platform och Native

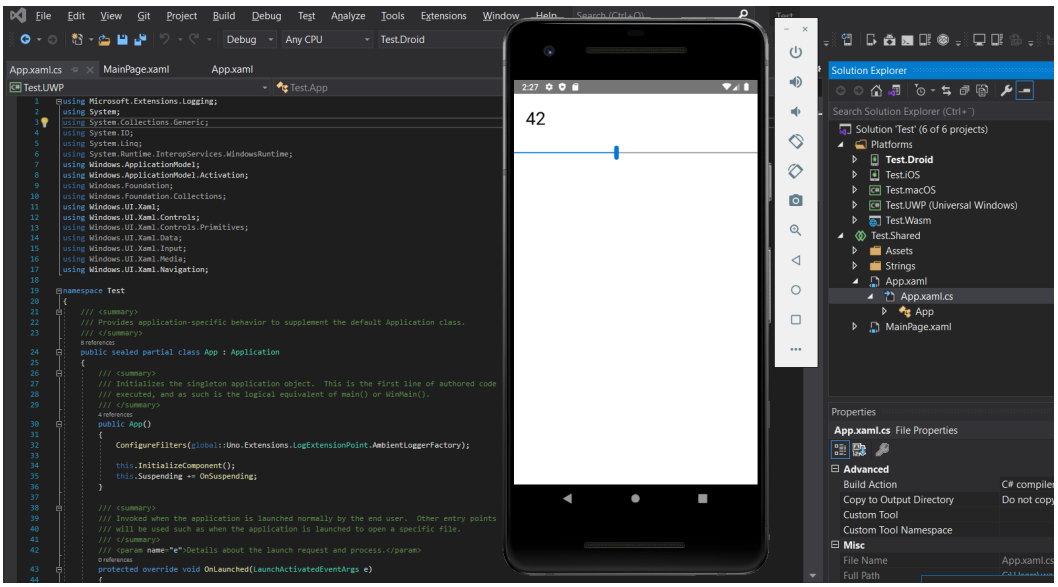
Det finns så kallade nativeapplikationer, vilket är applikationer som har utvecklats för ett specifikt operativsystem som till exempel Android eller iOS [2]. Vill man därefter göra det möjligt för applikationen att även köras i ett annat operativsystem måste man i princip utveckla en helt ny applikation. Detta resulterar i flera kodbaser, en för varje operativsystem som applikationen ska köras på.

En av nackdelarna med att skriva flera kodbaser är att det är tidskrävande eftersom koden som har skrivits för ett operativsystem kan inte återanvändas för ett annat operativsystem. Detta innebär att det blir mer kostsamt eftersom företaget behöver betala för flera utvecklare om ett företag har flera team av utvecklare där varje team ansvarar för utvecklingen för ett operativsystem. Fördelarna med native är att man har tillgång till alla verktyg och API:er som operativsystemet har att erbjuda. Native applikationer brukar även ha bättre prestanda än cross-platform applikationer. En annan fördel är att native applikationer brukar vara mer skalbara [2].

Därför finns också möjligheten att utveckla applikationer med hjälp av cross-platform-ramverk. Dessa ramverk möjliggör att man kan utveckla en kodbas som ska sedan fungera i flera operativsystem [2]. Det finns både för- och nackdelar med båda typerna av applikationsutveckling.

2.2 Visual Studio

För att kunna utveckla en prototyp som använder Uno-ramverket har Visual Studio använts. Visual Studio är en integrerad utvecklingsmiljö som utvecklades av Microsoft och släpptes 1997. Främst används Visual Studio för mjukvaruutveckling. Det kan vara allt från att skapa en hemsida till att utveckla en fullständig applikation som kan ge nytta i samhället. Programmeringsspråken som finns tillgängliga att använda när man utvecklar med Visual Studio är C++, C#, Python, Java och Node.js. Visual studio kan användas på Windows, macOS samt Linux. För att Uno-ramverket ska kunna användas i Visual Studio krävs det att Uno-ramverket installeras som tillägg. Figur 1 visar hur Visual Studio såg ut under utvecklingen.

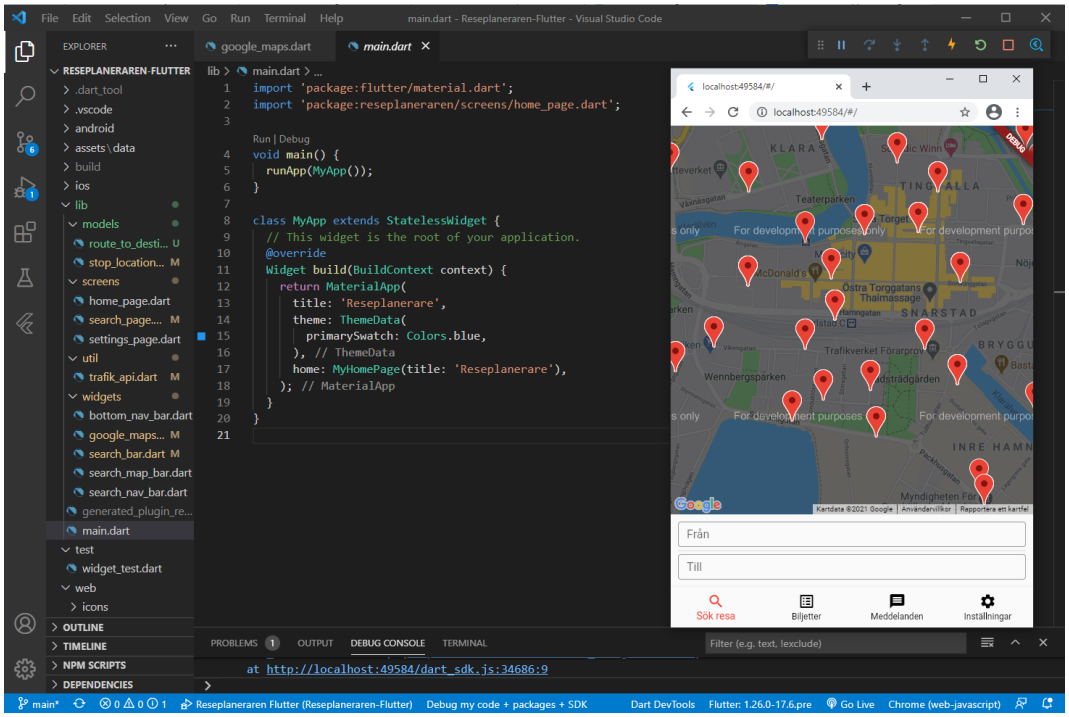


Figur 1. Bild på Visual Studio 2019 under utvecklingen.

2.3 Visual Studio Code

För att utveckla prototypen med Flutter kommer koden att skrivas i Visual Studio Code. Det är en lättviktig textredigerare som är open source och utvecklas av Microsoft, det initiala utsläppet av textredigeraren skedde den 29 april 2015.

Visual Studio Code kan köras på Windows, macOS och Linux och har inbyggt stöd för flera programmeringsspråk som TypeScript, JavaScript och Node.js. Det finns även stöd för andra språk vilket man får tillgång till genom att installera tillägg för språket. För att utveckla Flutter-applikationer behöver man installera tilläggen för Flutter och Dart. Figur 2 visar hur Visual Studio Code såg ut under utvecklingen.

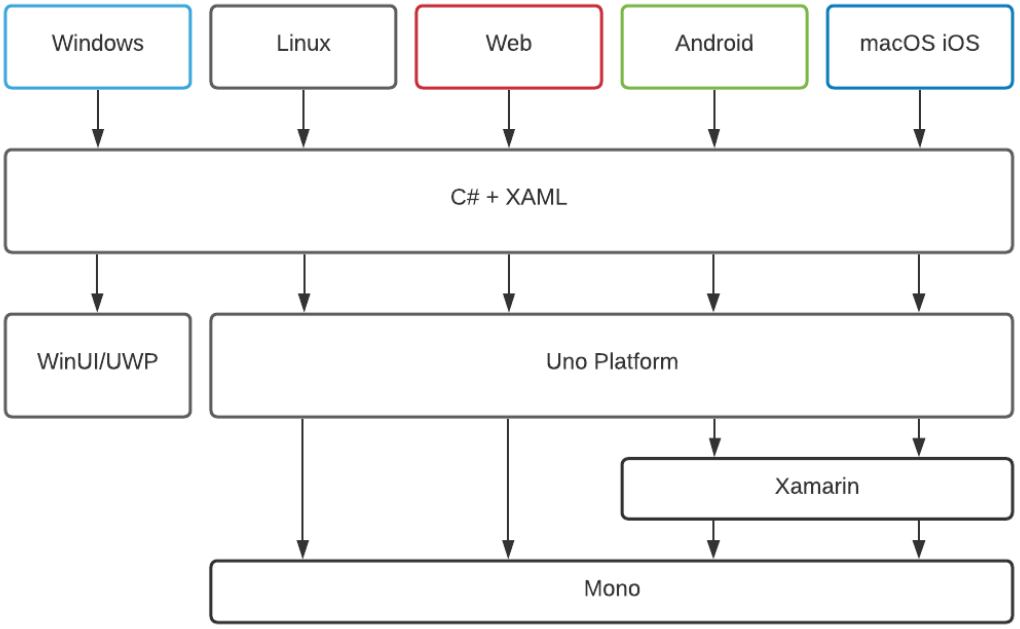


Figur 2. Bild på Visual Studio Code under utvecklingen

2.4 Uno Platform

Uno Platform är ett öppen källkod cross-platform-ramverk som släpptes först under maj månad år 2018. Den är ganska ny vilket har sina för- och nackdelar. Programmeringsspråk som används för att utveckla applikationer i Uno Platform är C# och XAML. I Uno Platform kan man utveckla applikationer för operativsystemen iOS, macOS, Linux, Android, man kan även utveckla webbapplikationer [4].

Vid utveckling av applikationer i iOS och Android använder Uno Platform sig av Xamarin Native Stack. Vid utveckling av webbapplikationer krävs det att man istället använder sig av WebAssembly (Wasm). För att webbapplikationerna ska renderas på webbläsaren används Uno Web Bootstraper som utnyttjar Mono-Wasm. Med hjälp av Uno Platform kan också Universal Windows Platform appar utvecklas. [5]. En bild på kopplingarna mellan plattformarna, operativsystemen och ramverken finns att se i figur 3. Bilden ritades med inspiration av bilden på Uno Platforms officiella hemsida [21].



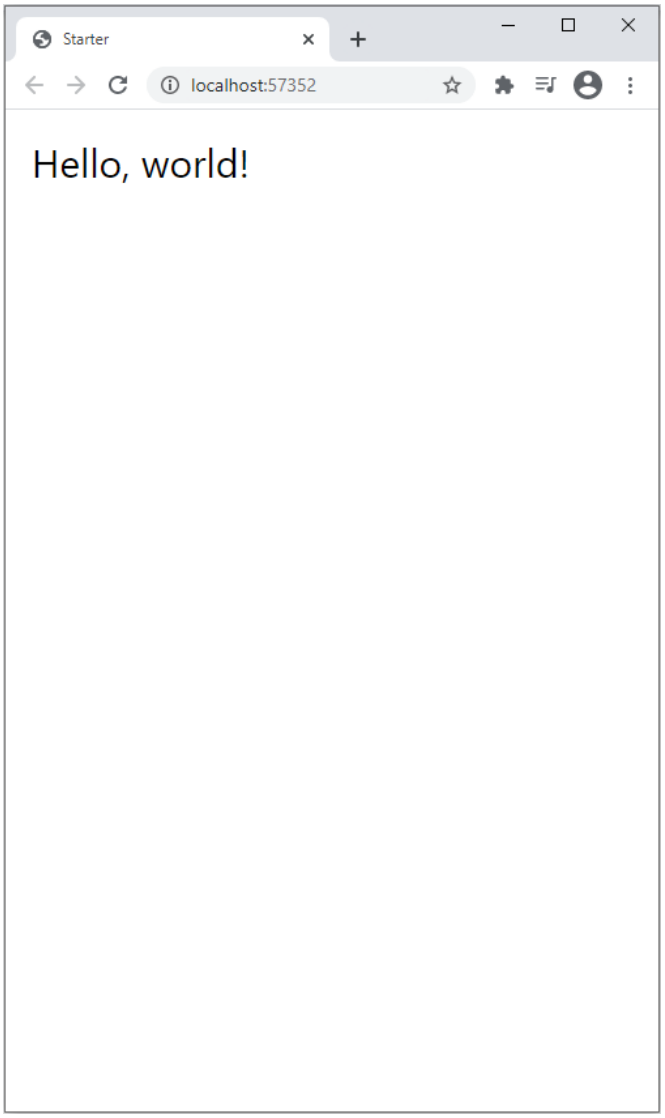
Figur 3. Övergripande arkitektur på hur Uno Platform fungerar.

I figur 5 finns en skärmdump på en applikation och i figur 4 visas koden för applikationen. Det är en enkel Hello World-applikation. Applikationens användargränssnitt definieras med hjälp av XAML-kod. Det finns en Grid-panel, vilket är en enkel panel som innehåller andra komponenter. Inuti Grid finns det en TextBlock där texten som skrivs ut på skärmen finns.

```
<Page
    x:Class="Starter.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Starter"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <TextBlock Text="Hello, world!" Margin="20" FontSize="30" />
    </Grid>
</Page>
```

Figur 4. Bild på XAML koden i Uno Platform



Figur 5. Bild på start applikationen i Uno Platform

2.5 Flutter

Flutter är ett cross-platform-ramverk som har utvecklats av Google. Ramverket släpptes maj 2017 och var då i version alpha. Den första stabila versionen släpptes 4 december 2018 [6]. Det är alltså ett relativt nytt ramverk.

Utöver iOS och Android har även Flutter stöd för webbapplikationsutveckling, däremot är det inte lika etablerat som stödet för de mobila operativsystemen. Stödet för webbapplikationsutveckling befinner sig i beta. Flutter använder dart2js för att kompilera koden till en enda JavaScript-fil. Dart2js används när man bygger en releaseversion av applikationen. Under utvecklingen när man debuggar

och testar applikationen används istället dartdevc [7]. I Flutter används programmeringsspråket Dart till skillnad från C# i Uno Platform. Det är inte ett av de mest populära programmeringsspråken men språkets syntax är ganska likt Java.

När det kommer till webbapplikationer har man två alternativ när det kommer till hur applikationen ska renderas på webbläsaren. Man kan använda HTML eller CanvasKit. HTML använder sig av HTML element, CSS, Canvas element och SVG element. CanvasKit använder sig av Skia som kompileras till WebAssembly och renderas med hjälp av WebGL [8].

Vilken renderingsmotor man väljer beror på utvecklarnas prioritering. Vill man hellre prioritera en mindre nedladdningsstorlek än prestanda bör man välja HTML. Prioriterar man hellre prestandan och att utseendet ser likadant ut på flera webbläsare bör man välja CanvasKit. Standarden i Flutter är att HTML används för mobiler och CanvasKit för stationära datorer [8].

En applikations utseende definieras i Flutter av komponenter som kallas för widgets. Man nästlar widgets med andra widgets, varje widget har en så kallad föräldra-widget. Det finns en root widget som hela applikationen utgår ifrån. Det bildas en hierarkisk struktur av widgets [9].

För att förstå den hierarkiska strukturen bättre finns bilder nedan på applikationen som skapas när man skapar ett nytt Flutter projekt, både koden och en skärmdump på hur applikationen ser ut när den körs, se figur 6 och 7. Några rader kod som inte är essentiella för detta exempel har tagits bort. Applikationen startas i metoden main på rad 2 där MyApp anropas. MyApp består av applikationens root widget, själva widgeten byggs av metoden Widget build(BuildContext context) på rad 7. Denna returnerar widgeten MaterialApp som i sin tur innehåller en annan widget i form av MyHomePage på rad 13. MyHomePage returnerar widgeten Scaffold. Scaffold består av flera widgets, en appbar, en Center widget i body på rad 25 som centrerar widgeten som den själv innehåller, en Column widget vilket ligger inne i Center widgeten och en Text widget som ligger inne i Column där texten i mitten av skärmen skrivs. Det finns även en FloatingActionButton widget, vilket är knappen i nedre hörnet som i sin tur innehåller widgeten Icon som anger hur knappen ska se ut.

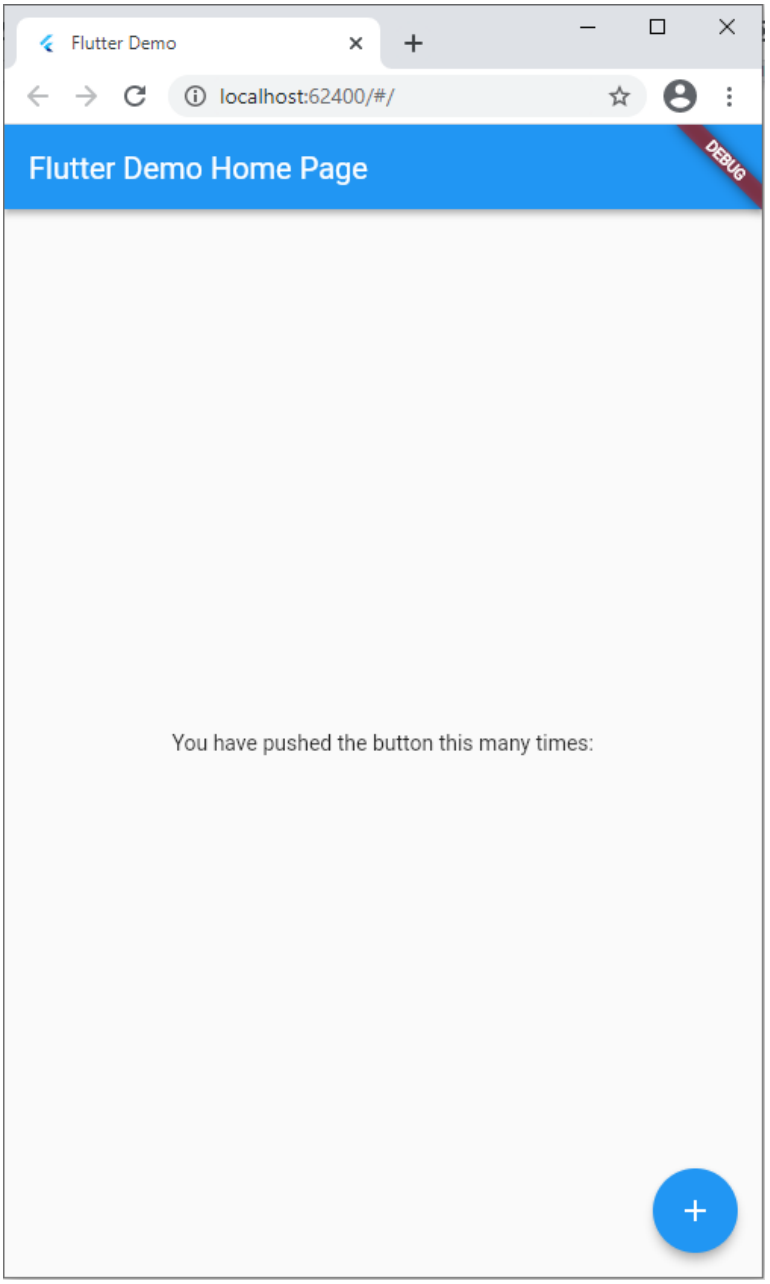
```dart
import 'package:flutter/material.dart';

Run | Debug
void main() => {runApp(MyApp())};

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ), // ThemeData
      home: MyHomePage(),
    ); // MaterialApp
  }
}

class MyHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Flutter Demo Home Page'),
      ), // AppBar
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text(
              'You have pushed the button this many times:',
            ), // Text
          ], // <Widget>[]
        ), // Column
      ), // Center
      floatingActionButton: FloatingActionButton(
        onPressed: null,
        tooltip: 'Increment',
        child: Icon(Icons.add),
      ), // FloatingActionButton
    ); // Scaffold
  }
}
```

Figur 6. Bild på koden av Flutter applikationen.



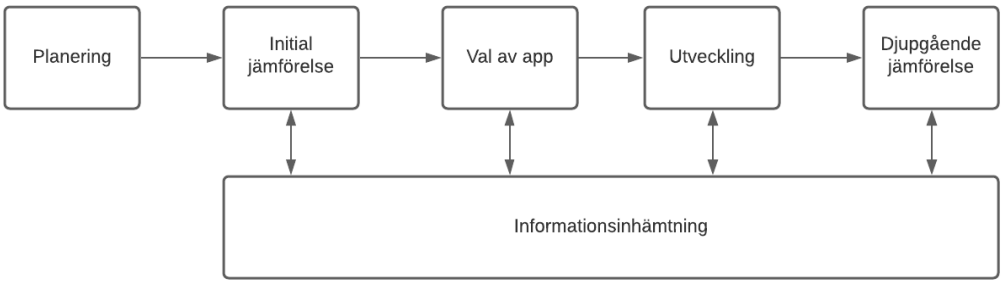
Figur 7. Bild på Flutter applikationen.

3 Metod

För att kunna utföra utvärderingen av ramverken på ett systematiskt sätt har examensarbetet delats in i sex faser. Arbetet delades in i följande faser.

1. Planering av examensarbetet
2. Informationsinhämtning.
3. Initial jämförelse
4. Val av app som ska utvecklas
5. Utveckling av prototyper
6. Djupgående jämförelse

Fas 1, 3, 4, 5 och 6 genomfördes sekventiellt. Enda fasen som inte genomfördes sekventiellt var fas 2, informationshämtning. Fas 2 genomfördes parallellt med fas 3, 4, 5 och 6. Anledningen till detta var att information behövde hämtas under större delen av examensarbetets gång. Figur 8 visar hur arbetsprocessen med de olika faserna gick till.

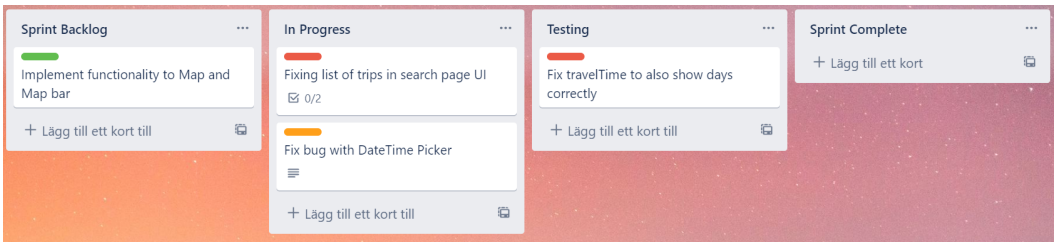


Figur 8. Flödesschema över arbetsprocessen.

3.1 Projektmodell

Under utvecklingen användes en informell version av Scrum [10]. Inga dagliga standup möten hölls. I början av varje sprint delades uppgifterna som skulle göras i mindre delar. Typiska roller som ingår i Scrum som Scrum Master, produktägare och utvecklingsteam tilldelades inte till olika personer, utan examensarbetarna hade ett gemensamt ansvar över allt. Vidare gjordes ingen tidsestimering för varje uppgift som skulle utföras. Uppgifterna som behövde göras blev tilldelade jämnt bland examensarbetarna, däremot hjälpte examensarbetarna varandra när det behövdes. För att implementera scrumtavlan användes hemsidan Trello [11]. Tavlan delades in i fyra sektioner: Sprint Backlog, In Progress, Testing och Sprint Complete. Varje uppgift som skulle göras tilldelades en färgad etikett som tyder på vilken prioritet de har. Grön innebär låg prioritet, orange innebär medel

prioritet och röd innebär hög prioritet. Eftersom möte med företagets handledare skedde en gång i veckan, sågs varje vecka som en sprint. Figur 9 visar hur tavlan kunde se ut under en sprint.



Figur 9. Bild på scrumtavlan i Trello

3.2 Planering

Examensarbetet inleddes med att planera arbetet fram till slutredovisningen och opponeringen. Hänsyn togs till tentaperioder. För att kunna presentera planeringen på ett bra och tydligt togs det fram ett Gantt-schema där arbetstiden delades in i olika kategorier som lades ut i Gantt-schemat.

Eftersom Gantt-schemat togs fram innan arbetet hade satts igång ordentligt uppstod vissa svårigheter med att förutse hur länge respektive fas bör ta. Men trots det fick man fram en rimlig tidsplanering som man kan utgå ifrån. Delar av tidsplaneringen kunde ändras beroende på hur arbetet verkligen fortskred, men det fanns vissa deadlines som inte gick att förskjuta.

3.3 Informationsinhämtning

Genom hela examensarbetet gjordes en hel del informationsinhämtning. Första tillfället då informationsinhämtningen gjordes var för att kunna utföra den initiala jämförelsen där det undersöktes vilket eller vilka ramverk som skulle ingå i den djupgående jämförelsen.

För att kunna göra den initiala jämförelsen behövdes information angående olika ramverk. Denna information inhämtades främst genom att läsa på ramverkens officiella dokumentation, andra källor på nätet användes också. Olika artiklar som går igenom vilka ramverk kan vara bra att lära sig lästes. Detta för att först kunna välja ut de mest intressanta ramverken som sedan ska ingå i den initiala jämförelsen. Alla ramverk som existerar kan inte ingå i den initiala jämförelsen, därför valdes det ut ett antal ramverk som ansågs vara intressanta.

För att hämta information för att kunna jämföra ramverken utifrån kriterierna i den initiala jämförelsen användes ett antal källor. För att jämföra populariteten användes främst GitHub och Stack Overflow då de ger en ganska bra bild på hur många utvecklare som använder respektive ramverk. Popularitet i detta sammanhanget innebär hur många utvecklare som använder ramverken, detta korrelerar med vilka möjligheter det finns att få hjälp på internet. Det finns inte någon anledning att betvivla att siffrorna som de anger inte stämmer då de är trovärdiga källor som är populära inom utveckling. Angående de andra kriterierna som handlar om att stödja utveckling av webbapplikationer, om ramverken är gratis och hur nyligen ramverken kom ut, hittades informationen främst på respektive ramverks officiella hemsida. Endast då informationen inte kunde hittas på deras hemsidor letades informationen upp i andra källor.

3.4 Initial jämförelse

För att kunna genomföra den initiala jämförelsen som har syfte att bestämma vilka eller vilket ramverk som ska jämföras med Uno Platform bestämdes det först vad jämförelsen skulle baseras på. Detta krävde att examensarbetarna tog fram lämpliga kriterier som skulle användas för att jämföra ramverken i den initiala jämförelsen.

Då Softhouse är mest intresserade av cross-platform-ramverk där man inte behöver bygga något i native, måste man se till att potentiella ramverk kan utveckla applikationer med icke nativelösningar. Ramverken måste stödja utveckling av webbapplikationer. Något annat att ta hänsyn till är, när ramverket introducerades. Äldre ramverk som fortfarande inte är populära, är inte lika intressanta då det brukar finnas en anledning till varför de inte är populära. Nya ramverk är potentiellt mer intressanta att undersöka då det inte finns lika mycket information om dem. Men det finns även nackdelar med nya ramverk, till skillnad från populära och väletablerade ramverk brukar nya ramverk ha många buggar. Dessutom finns det inte lika mycket tillgänglig hjälp och information för nya ramverk. Detta leder till ett annat kriterium som handlar om vilka möjligheter det finns att få hjälp på internet. Om ramverken är gratis ansågs också vara ett lämpligt kriterium att ha med.

Detta kan sammanfattas till kriterierna:

- Vilka möjligheter finns det för att få hjälp på internet?
    - Hur populärt är ramverket?
- När introducerades ramverket?

- Kan man utveckla webbapplikationer som automatiskt anpassar sig efter vilket operativsystem som enheten körs på utan att använda nativelösningar?
- Är det gratis att använda ramverket?

Kriterierna valde examensarbetarna ut. Efter att kriterierna hade valts ut genomfördes undersökningen av olika ramverk. Det resulterade i att ramverket Flutter valdes som ramverket som ska ingå i den mer djupgående jämförelsen med Uno Platform. Valet av Flutter verifierades med företaget och de höll med valet.

3.5 Val av app som ska utvecklas

Efter att den initiala jämförelsen hade gjorts och ramverken som ska undersökas mer ingående hade bestämts fattades det beslut om vilken sorts applikation som ska utvecklas. Detta gjordes tillsammans med handledaren på Softhouse. Softhouse är företaget som har utvecklat Skånetrafikens applikation och även andra reseplanerare. Därför var det intressant för vår handledare om det skulle kunna utvecklas någon sorts reseplanerare. Eftersom han själv jobbar med Skånetrafiken i företaget. Företaget var däremot även öppet för andra förslag men då utvecklingen av reseplanerare är av stort intresse för företaget ansågs det vara rimligt att undersöka ramverken genom att utveckla en reseplanerare.

Eftersom utvecklingen gjordes inom ramen för ett examensarbete behövdes det göras ett antal avgränsningar. Prototyperna som utvecklades behövde inte ha med all funktionalitet som finns i olika reseplanerare som finns ute i marknaden idag.

3.6 Utveckling av prototyper

Efter att det hade bestämts vilken sorts prototyp som skulle utvecklas kunde utvecklingen sättas igång. De två olika prototyperna utvecklades inte parallellt utan fokus låg på en prototyp i taget. Detta ansågs av examensarbetarna vara det bästa alternativet.

Den första prototypen utvecklades med Flutter. För att utveckla Flutter-prototypen användes textredigeraren Visual Studio Code. Den information om ramverket som behövdes under kodningen fanns tillgänglig på olika webbplatser. Främst användes officiell dokumentation på ramverkets hemsida. Flutter har en mängd information på sin hemsida, det finns olika handledningar där de går igenom hur man implementerar viss funktionalitet. Det finns även många videor där de förklarar hur olika widgets fungerar. Ofta räckte det med att studera den officiella

dokumentationen men då och då användes även andra källor. Det kunde vara allt från att få inspiration från människor som lagt upp projekt på GitHub till att läsa på Stack Overflow.

Utvecklingen av Flutter-applikationen skedde inte utan vägledning utan handledaren från företaget hjälpte till. Innan utvecklingen av Flutter-applikationen påbörjades framförde handledaren på Softhouse att Flutter-applikationens utseende skulle efterlikna Skånetrafikens applikation.

För att möjliggöra testning av applikationen på olika mobiler lades applikationen ut på GitHub Pages [12]. Det är ett webbhotell som använder GitHub från vilket man kan hämta filerna som bygger upp hemsidan. GitHub Pages användes eftersom det var lätt att komma igång med och dessutom användes GitHub redan för applikationen, examensarbetarna ansåg därför att det var rimligt att använda GitHub Pages.

Under Flutter-utvecklingens gång hölls det möten kontinuerligt med både skolans handledare och handledaren från Softhouse. Varannan vecka hade examensarbetarna ett möte med skolans handledare där det diskuterades förbättringar av rapporten samt utvecklingen av prototyperna. En gång per vecka hölls ett möte med handledaren på Softhouse för att kunna få snabb respons från handledaren om förslag på förbättringar av prototypen och annat. En gång per månad hade också examensarbetarna ett möte med en chef från Softhouse eftersom chefen var intresserad av hur examensarbetet fortskred. Under mötet med chefen visades det som hade gjorts.

3.7 Djupgående jämförelse

Efter utvecklingen av prototyperna gjordes en djupgående jämförelse mellan Flutter och Uno Platform utifrån ett antal andra kriterier. Kriterierna valdes delvis genom en diskussion med handledaren på Softhouse, där handledaren berättade vad han tyckte var viktigt. Prestandan var kriteriet som handledaren var främst intresserad av. Andra kriterier valdes genom att göra en informationssökning angående vad andra jämförelser mellan ramverk har baserats på, men även vad examensarbetarna ansåg var rimligt att jämföra efter att ha utvecklat prototyperna.

Kriterierna som valdes var: programmeringsspråk, kod, storlek, kompileringstid, tid att hämta data från API, prestanda, hot reload, paket och tillgängligheten av information

Den djupgående jämförelsen genomfördes genom att examensarbetarna gick igenom kriterierna och jämförde ramverken utifrån de. Jämförelsen gjordes på två datorer för att se till att det inte uppstår stora skillnader beroende på dator. Gällande kriterierna där det mäts tid: kompileringstid och tid att hämta data från API, gjordes tidsmätningen på samma dator för att rättvist kunna jämföra tiderna. När ett kriterium utvärderades med tidmätning gjordes det 10 försök för ett noggrant resultat. Det som var mest intressant när det gällde jämförelsen var själva prestandan och även upplevelsen av att utveckla applikationerna med ramverken.

3.8 Källkritik

Anledningen till varför källorna som använts i examensarbetet ansågs vara trovärdig kommer att beskrivas här.

Källa: [1], [10]
Hemsidorna tillhör företaget som examensarbetet görs i samarbete med. Företaget är ganska känt i Malmö där de bland annat har/haft samarbete med Skånetrafiken. Vidare måste Softhouse vara en god källa till hur Softhouses system fungerar. På så vis är källan trovärdig.

Källa: [2]
Hemsidan tillhör ett välrenommerat företag inom området IT. Källan listar upp vilka klienter företaget har och vilka priser de har vunnit. Det finns även en lista med olika applikationer som företaget har utvecklat, de har utvecklat appar för företag som Dollar Shave Club, Aspiration och Sprent. På hemsidan Clutch har företaget 4,9 stjärnor av fem från 27 recensioner. Där recensionerna har gjorts av andra företag som har varit tidigare klienter. På grund av dessa anledningar anser vi att källan är trovärdig.

Källa: [3], [4], [5], [6], [7], [8], [9], [16], [21], [23], [24], [27] och [30]
Dessa källor tillhör de olika ramverkens officiella hemsidor. Eftersom de är officiella hemsidor kan de anses som trovärdiga. Dessutom har informationen visat sig att stämma när man använt sig av infon.

Källa: [11]
Trellos officiella hemsida. Användes för scrumtavlan, ingen övrig information hämtades från källan. Trello finns också som en mobilapplikation, i Google Play har den 4,5 stjärnor av fem från över 102,000 recensioner. På grund av Trellos

popularitet och att hemsidan endast användes för scrumtavlan fanns det ingen oro över trovärdigheten.

Källa: [12]
GitHub Pages officiella hemsida. Användes för att kunna testa prototyperna, ingen övrig information hämtades från källan.

Källa: [13], [17], [20] och [25]
GitHub en populär sida som många utvecklare känner till och använder. Det finns över tre miljoner organisationer samt 65 miljoner utvecklare som använder sig av GitHub. Den enda informationen som hämtades från GitHub var hur många stjärnor ramverken hade och vilket år React först introducerades. Endast statistik hämtades. På grund av dessa anledningar anser vi att källan är trovärdig.

Källa: [14], [15], [19], [26], [31] och [32]
Stack Overflow är en populär hemsida som har runt 14 miljoner registrerade användare. Bara siffran 14 miljoner tyder på trovärdigheten av källan. Hade inte källan varit trovärdig hade det nog inte varit många som hade använt källan. Dessutom har källan fått över 21 miljoner frågor som besvarats med 31 miljoner svar. Den enda informationen som hämtades från Stack Overflow var hur många frågor det fanns samt hur många svar till frågorna.

Källa: [18]
Hemsidan har mycket information inom IT som överensstämmer med andra källor. Artikeln som det hämtades information ifrån har besökts 756,200 gånger och det anges vem skribenten är. Han har skrivit över 88 artiklar på hemsidan och sammanlagt har de över 29,7 miljoner visningar. Dessutom är han en certifierad Microsoft MVP vilket ges till experter som delar deras kunskap med andra. Tittar man på hans Linkedin kan man se att han har gått på en utbildning inom datavetenskap och har en hel del erfarenheter. På grund av dessa anledningar anser vi att källan är trovärdig.

Källa: [22]
Javatpoint är en hemsida som innehåller information om olika programmeringsspråk och andra teknologier. Vidare ges erbjudandet om att få träning om man vill öka sin kunskap inom något som källan erbjuder. På hemsidan finns det information om ägaren där det bland annat står vilken utbildning han har slutfört. Kontaktinformation finns också. Det finns hemsidor där man kan skriva recensioner av andra sidor, på hemsidan Sitejabber har Javatpoint fått fem av fem stjärnor från 27 recensioner, på Mouthshut har

Javatpoint fått 3,96 stjärnor av fem från 775 röster. Dessutom överensstämmer informationen som hämtats från hemsidan med andra källor. På grund av dessa anledningar anser vi att källan är trovärdig.

Källa: [28]
Ett exempel på en Ionic applikation, ingen övrig information hämtades från hemsidan.

Källa: [29]
Officiell hemsida för Flatpickr-komponenten som användes i Uno Platform. Komponenten användes och det fungerade.

Källa: [33]
pub.dev är den officiella sidan där alla Flutter-paket hämtas. Paketen som användes hämtades från sidan och de fungerade. Informationen som hämtades från pub.dev var hur många paket som finns tillgängliga i Flutter.

Källa: [34]
nuget är den officiella sidan där alla Uno Platform-paket hämtas. Paketen som användes hämtades från sidan och de fungerade. Informationen som hämtades från nuget var hur många paket som finns tillgängliga i Uno Platform.

4 Analys

Det här kapitlet handlar om den initiala jämförelsen som gjordes under detta examensarbetet.

4.1 Initial jämförelse

Den initiala jämförelsen utgår utifrån dessa kriterier:

- Vilka möjligheter finns det för att få hjälp på internet?
  - Hur populärt är ramverket?
- När introducerades ramverket?
- Kan man utveckla webbapplikationer som automatiskt anpassar sig efter vilket operativsystem som enheten körs på utan att använda nativelösningar?
- Är det gratis att använda ramverket?

Olika cross-platform-ramverk undersöktes och utifrån dessa kriterier bestämdes vilket ramverk som ska jämföras med Uno Platform. Undersökningen av de olika cross-platform-ramverken utfördes genom att examensarbetarna först undersökte vilka cross-platform-ramverk som fanns. Namnen på ramverken skrevs ner i ett dokument. Sedan utfördes en kortare undersökning av varje ramverk som fanns i dokumentet för att få en bättre uppfattning av ramverken. Undersökningen gick ut på att hitta ramverken med störst potential utifrån kriterierna. Tre ramverk valdes ut.

Dessa ramverk är Flutter, React.js och Ionic, och de undersöktes ingående utifrån kriterierna ovan. Ett exempel på ett annat ramverk som undersöktes men som inte ansågs vara lika intressant som de tre som undersöktes är ramverket Blazor. Likt Uno Platform utnyttjas WebAssembly för att skapa applikationer. Blazor är ett relativt nytt ramverk som ser intressant ut, men det föredrogs att jämföra Uno Platform med andra ramverk som inte använder sig av WebAssembly och C# vilket Blazor gör.

4.1.1 Flutter

När det gäller populariteten hos ramverket vilket hör ihop med vilka möjligheter det finns för att få hjälp på internet så är Flutter relativt populärt. Ramverket har ca 120,000 stjärnor på GitHub [13] och det finns cirka 86,624 frågor i Stack Overflow om Flutter [14]. Enligt den senaste årliga undersökningen (2020) av Stack Overflow, används Flutter av 7.2% av de som svarade på undersökningen.

Detta resultat var under rubriken “Other Frameworks, Libraries, and Tools” där utvecklarna anger vilka verktyg de använder. När det frågas efter “mest gillade” ramverk, verktyg och bibliotek ligger Flutter på tredje plats med 68.8%. Detta innebär alltså att Flutter är ett ramverk som utvecklare, enligt Stack Overflow, gillar att använda [15].

Det finns en lista med olika applikationer som har utvecklats med hjälp av Flutter i Flutters GitHub sida [16]. Med hjälp av dessa applikationer kan man få en inblick i Flutters potential. Bland annat kan man se hur applikationens användargränssnitt anpassar sig till vilket operativsystem applikationen körs på.

Sammanfattningsvis kan man säga att Flutter är ett gratis och relativt nytt ramverk och man kan utveckla applikationer som fungerar för de olika operativsystemen utan att använda nativelösningar. För att vara ett nytt ramverk finns det goda möjligheter att få hjälp på internet.

4.1.2 React.js

React.js är ett JavaScript-bibliotek med öppen källkod som utvecklades av företaget Facebook under årtalet 2013. Det är alltså relativt gammalt då det först introducerades för åtta år sedan [17].

Biblioteket kan användas för att skapa användargränssnitt för alla möjliga sorters applikationer som använder endast en enda webbsida i taget när applikationen körs. Webbsidan som körs i applikationen uppdateras dynamiskt och gör det möjligt att uppdatera data på webbsidan utan att behöva ladda om webbsidan. Implementationen av React.js kan ske med JavaScript XML(JSX). JSX är förlängning till JavaScript som gör det möjligt att inkludera HTML-kod som sedan används för att rendera React komponenter [18].

Angående möjligheterna för att få hjälp på internet är de ganska stora. Under år 2020 var React.js den näst mest gillade webbramverket bland världens utvecklare enligt en undersökning gjord utav Stack Overflow [15]. Det finns över 304,644 frågor i Stack Overflow om React,js [19], vilket visar hur lätt det är att hitta information om React.js. Vidare har React.js 168,000 stjärnor på GitHub [20]. Det är lätt att hitta information om React.js helt enkelt.

Det finns en lista med olika projekt som har utvecklats med hjälp av React.js. Några exempel är Emoji Search, Snap Shot och BMI calculator. Listan finns i den officiella hemsidan av React.js [21].

Sammanfattningsvis går det att använda React.js för att utveckla användargränssnitt för applikationer med olika operativsystem. React.js är ett äldre ramverk men är å andra sidan välkänt vilket gör det lätt att hitta information vilket är en fördel. Ramverket är även gratis att använda.

4.1.3 Ionic

Ionic är ett cross-platform-ramverk som används för att utveckla både mobilapplikationer och webbapplikationer. Ramverket introducerades i slutet av 2013 och är således ett relativt gammalt ramverk. Däremot har det varit med om en del förändringar, när Ionic kom ut kunde man endast utveckla mobilapplikationer med hjälp av Angular. Det fanns inget stöd för utvecklingen av webbapplikationer. År 2017 tillkom det stöd för webbapplikationer men det fanns fortfarande endast stöd för Angular [22].

Ramverket utnyttjar webbteknologier som HTML, CSS och JavaScript. Ionic har sedan 2019 även stöd för flera JavaScript-ramverk som Angular, React och Vue [23].

Man kan utveckla nativeapplikationer för Android och iOS vilket inte är intressant för detta examensarbete. När det gäller stödet för webbapplikationer utvecklar man så kallade Progressive Web Apps (PWA). Dessa är webbapplikationer som använder modern webbteknologi. Applikationerna ska fungera utmärkt på både mobilen och datorn. Med Ionic kan man även bygga sina applikationer både som PWA och nativeapplikationer samtidigt [24]. För detta examensarbete är PWA det som är mest intressant med Ionic.

När det gäller populariteten av ramverket, har Ionic cirka 44,097 stjärnor i GitHub [25]. I Stack Overflow finns det 47,010 frågor angående Ionic [26]. Över 10 miljoner applikationer har utvecklats med ramverket, det finns även över 5 miljoner utvecklare som använder Ionic. Detta innebär att man inte borde ha problem med att hitta hjälp på internet gällande ramverket [27].

Här är ett exempel på en webbapplikation som har utvecklats med hjälp av Ionic [28].

Sammanfattningsvis kan man säga att Ionic är ett gratis och gammalt ramverk som har undergått en del förändringar under åren, och man kan utveckla applikationer som fungerar för de olika operativsystemen utan att använda

nativelösningar genom PWA. Dessutom finns det goda möjligheter att få hjälp på internet om det behövs.

4.1.4 Slutsats

Efter att ha undersökt olika ramverk togs det ett beslut av examensarbetarna som också verifierade att Softhouse höll med valet angående vilket ramverk som ska ingå i den djupgående jämförelsen. React.js uteslöts då Softhouse som ett företag har mycket erfarenhet av ramverket. Detta medför att det inte är lika intressant att göra en mer ingående undersökning när det gäller just React. Det återstår att välja mellan Flutter och Ionic. De två ramverken har båda ett stöd för webbapplikationer vilket är viktigt för examensarbetet. Bland dessa två ramverk är Flutter det ramverk som har mest information ute på nätet angående ramverket. Gällande när dessa ramverk släpptes var Ionic först ute på marknaden, redan 2013 medan Flutter släpptes 2017. Trots att det är fyra år yngre har Flutter haft större framgång än Ionic. Angående språken som används i dessa två ramverk använder Flutter språket Dart som är mer likt Java vilket utvecklarna har tidigare erfarenhet av. Ionic å andra sidan använder sig av språken HTML, CSS och JavaScript. Efter en diskussion om vilket programmeringsspråk som föredrogs att arbeta med valdes Dart då det är ett språk som inte är lika välkänt. Detta ledde slutsatsen att välja Flutter att jämföra med Uno Platform, eftersom Flutter ansågs som ett bättre alternativ än Ionic enligt jämförelsen ovan.

Figur 10 visar en tabell som sammanfattar jämförelsen. Ramverken betygsattes av examensarbetarna utifrån en skala på 1 - 3 där 1 är bäst. Gällande kriterier där de antingen uppfylls eller inte innebär ett X att ramverket uppfyller kriteriet.

Ramverk	Popularitet	Ny	Webb	Gratis
React	1	3	X	X
Flutter	2	1	X	X
Ionic	3	2	X	X

Figur 10. Tabell som visar resultatet av den initiala jämförelsen.

5 Resultat

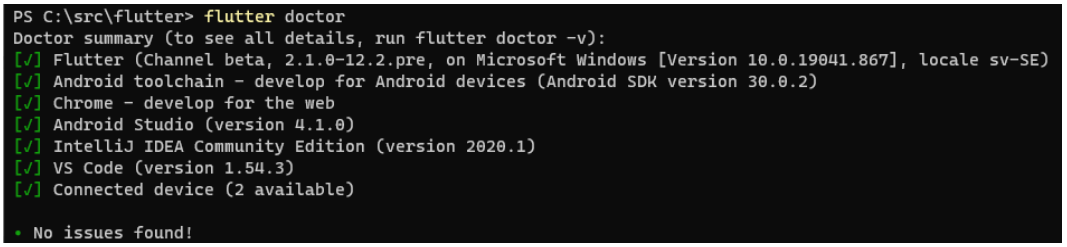
I detta kapitel beskrivs förberedelserna som gjordes av examensarbetarna innan det gick att sätta igång med utvecklingen av applikationerna. Därefter redogörs för utvecklingen av prototyperna. Sist presenteras resultatet av den djupgående jämförelsen mellan ramverken

.

5.1 Installation av Flutter

Nedan är instruktionerna som följdes för att få Flutter redo för utvecklingen. Installationen gjordes på en dator med Windows 10.

1. Flutters SDK-paket laddades ner från Flutters officiella hemsida.
2. Filen extraherades och "flutter" mappen placerades i C:\src\flutter.
3. Sökvägen i systemets miljövariabler uppdaterades genom att lägga till "C:\src\flutter\bin" i PATH.
4. Kommandon "flutter doctor" kördes i windows konsolen när man befann sig i "C:\src\flutter\" för att ta reda på om allt var redo eller om något annat behövdes installeras. I figur 11 visas hur det såg ut efter man kör kommandon.
5. Visual Studio Code laddades ner från Visual Studio Codes officiella hemsida. VSCode startades, sedan installerades Flutter pluginet genom att man först tryckte på View > Command Pallete och sedan valde man Extensions: Install Extensions och där valde man Flutter plugin.
6. För att skapa projektet skapades ett projekt direkt genom VSCode, men ett alternativ var att köra kommandon "flutter create namnPåApplikationen" i konsolen.



Figur 11. Bild på konsolen efter "flutter doctor" körs.

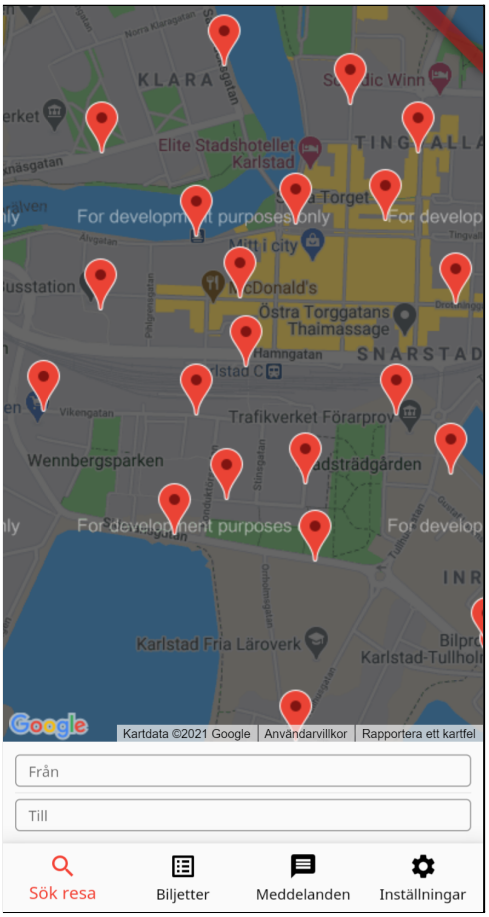
5.2 Installation av Uno Platform

Nedan är instruktionerna som följdes för att få Uno Platform redo för utvecklingen. Installationen gjordes på en dator med Windows 10.

1. Visual Studio laddades ner från Visual Studios officiella hemsida.
2. Universal Windows Platform, Mobile development with .NET och ASP.net and web behövdes installeras med Visual Studio.
3. Visual Studio startades och man tryckte på “continue without code”.
4. Man tryckte “extensions” och valde därefter “Online” fliken.
5. Man sökte efter “Uno” för att få upp “Uno Platform Solution Templates”, därefter valde man att ladda ner Uno Platform Solution Templates.
6. Visual Studio startades om och man valde “Create a new project”.
7. För att skapa ett Uno projekt sökte man “Uno” i “Search for templates” och valde sedan “Cross-Platform App (Uno Platform)” och tryckte på next.
8. Sist valde man namn och plats för projektet och tryckte sedan på “Create”.

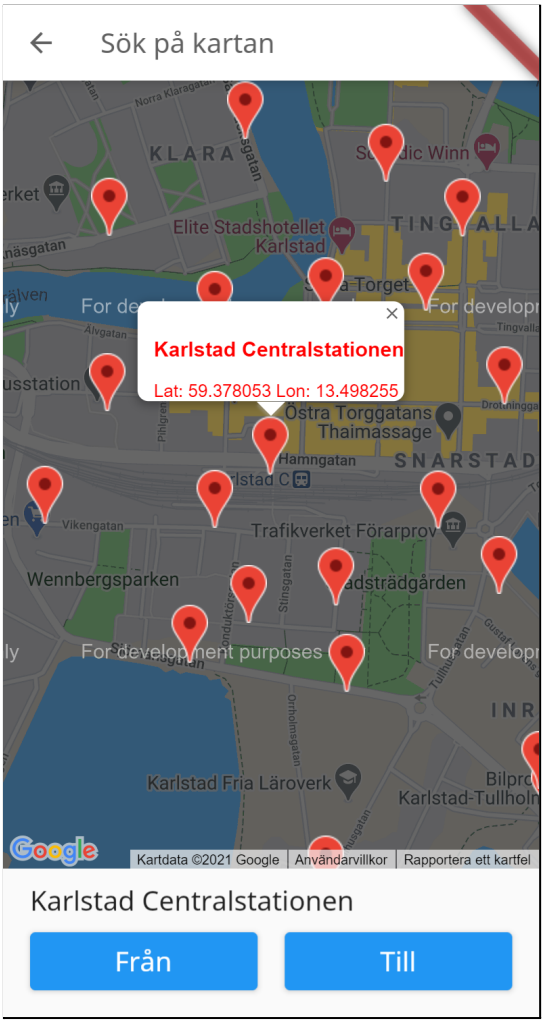
5.3 Utveckling med Flutter

För att testa utvecklingen i Flutter utvecklades en reseplanerare med flera olika funktioner. Första funktionen är kartan som finns på första sidan av Flutter-applikationen och som är till för att ge en bättre blick över området och hållplatserna som finns tillgängliga. Se figur 12. Det finns även en navigeringsmeny som består av två fält, “Från” och “Till” samt fyra olika knappar vars funktionalitet inte har implementerats. Trycker man på “Från” eller “Till” tas man vidare till nästa sida där man kan göra själva sökningen.



Figur 12. Bild på startsidan i Flutter applikationen.

Trycker man på kartan döljs navigationsmenyn och en ny meny kommer upp. Den nya menyn består av två knappar som används för att kunna välja hållplatsen man ska resa ifrån eller hållplatsen man vill nå. Man kan endast välja ett av dessa två alternativ från denna vyn. Det går alltså inte att bestämma både vart användaren ska åka ifrån samt var din destination är. Genom att trycka på en av markörerna som representerar en hållplats och sedan trycka på “Från” eller “Till” tas man vidare till nästa sida (Söksidan) där man sedan själv kan skriva in det andra alternativet som man inte valde. I figur 13 ser man hur det kan se ut när man trycker på en markör i sidan.



Figur 13. Bild på kartan i startsidan efter man har tryckt på en markör.

Söksidan har ett "Från" och "Till" fält där användaren skriver in namnen på ursprungsplatsen samt destinationen för att göra en resesökning. Namnet på en ursprungsplats eller destination som skrivs i "Från" eller "Till" fältet behöver vara minst tre bokstäver för att det ska dyka upp förslag. När man har valt ursprungsplats och destination får man upp en lista med alla resor som är möjliga att utföra. Varje resa visar starttiden och ankomsttiden för resan. Det kan även visas namnen på hållplatserna som besöks under resan och namnet på fortskaffningsmedlet, om de visas eller inte beror på antalet etapper som resan består av. I figur 14 kan man se hur resor med olika antal etapper visas.

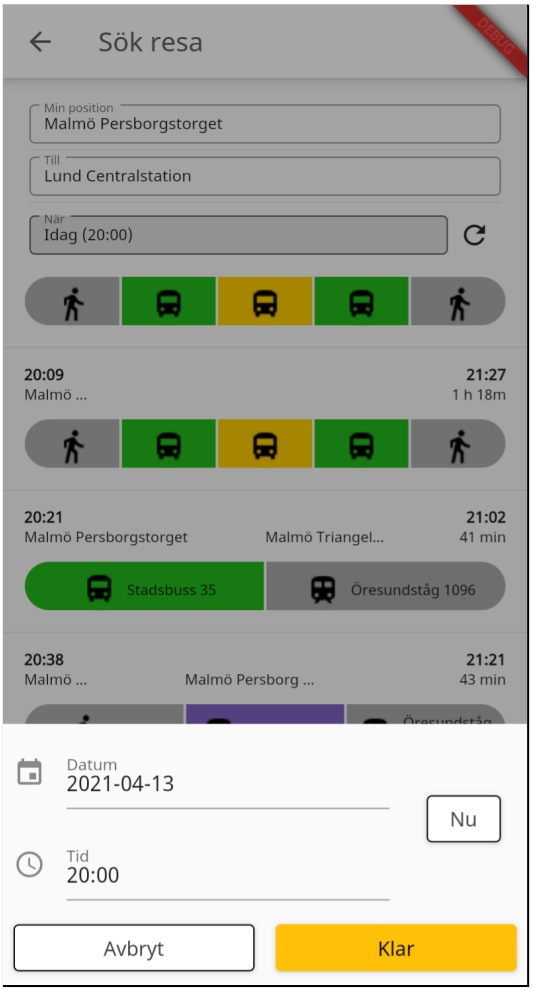
Vidare visas lämplig färg och ikon för resans olika delar. En bussresa visas genom en bussikon med grön bakgrundsfärg. En resa kan genomföras med en blandning av bussar och tåg och andra fortskaffningsmedel och också gång i några hundra meter. Gång och Öresundståg representeras av grå färg, pågatåg av lila färg,

regionbuss och taxi av gul färg, expressbussar av rosa färg, färjor av ljusblå färg och övriga tåg av blå färg.

Figur 14. Bild på Söksidan i Flutter applikationen.

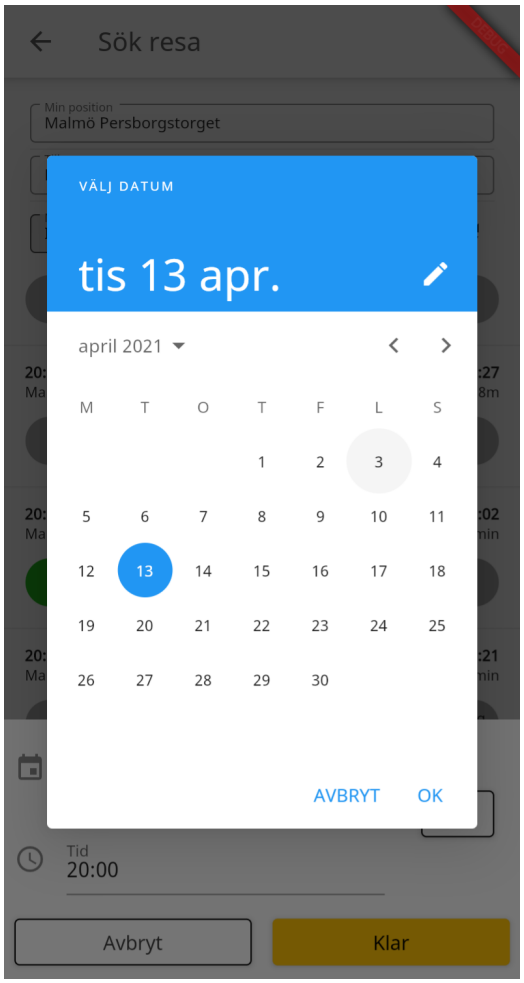
En annan funktion är att kunna söka efter en resa där ursprungsplatsen är koordinaterna för enheten som kör applikationen. Detta görs genom att låta "Från" fältet i söksidan vara tom.

Vidare implementerades det en funktion som gör det möjligt att filtrera efter datum och tid när man gör en resesökning. För att göra detta trycker man på "När" fältet för att få upp en meny. Menyn består av två fält, "Datum" och Tid" som visar vilket datum respektive tid som har valts. Det finns även tre knappar "Nu", "Avbryt" och "Klar". Trycker man på "Avbryt" göms menyn och inga ändringar sparas. Genom att trycka på "Klar" sparas ändringarna man har gjort på datum och tid och listan med resor uppdateras enligt de nya ändringarna. Om man trycker på "Nu" sätts datumet och tiden till enhetens nuvarande datum och tid vars applikationen körs. I figur 15 finns det en bild på menyn.

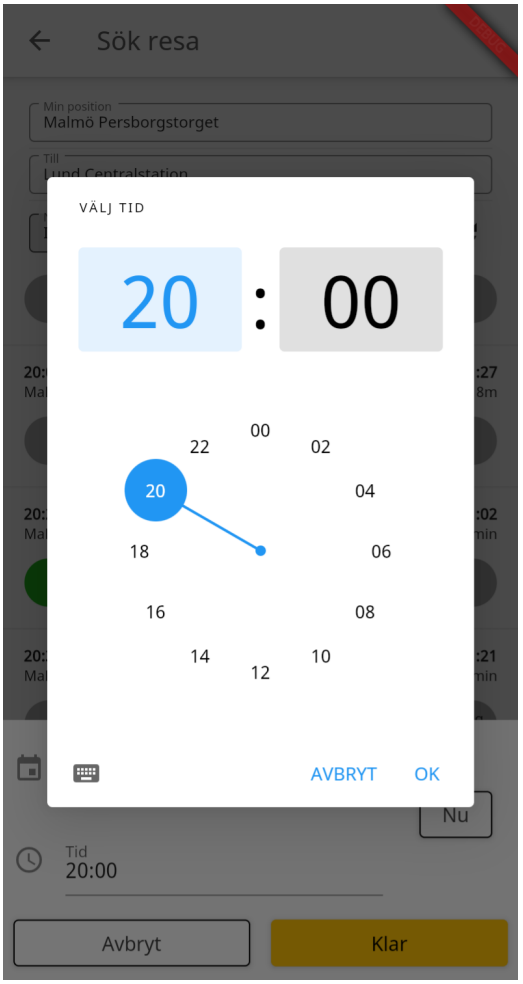


Figur 15. Bild på filtreringsmenyn.

När man trycker på “Datum” i menyn kommer det upp en kalender där man kan välja datum. I figur 16 visas det hur kalendern ser ut. Trycker man på “Tid” kommer det upp en klocka där man kan välja tid. I figur 17 visas det hur klockan ser ut.



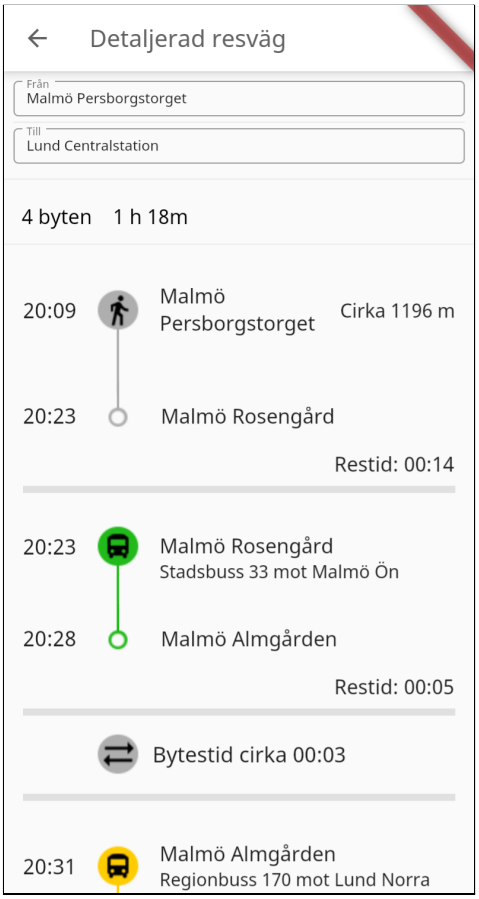
Figur 16. Bild på kalendern.



Figur 17. Bild på klockan.

För varje resa finns det funktionen att få en detaljerad resväg av resan om man önskar så. Man tas till den “Detaljerad resväg”-sidan genom att trycka på någon av resorna i söksidan.

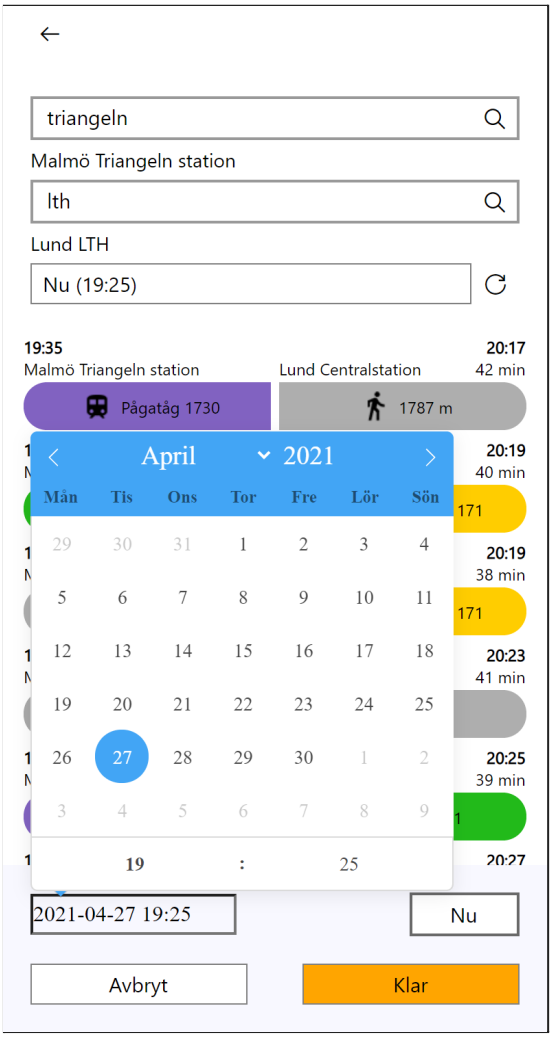
I “Detaljerad resväg”-sidan visas de olika transportmedlen som man behöver ta, under vilka tider, hållplatserna som man behöver besöka, bytestiden mellan hållplatser, totala andelen byten, totala restiden, restiden för samtliga etapper samt om det krävs att man behöver gå några hundra meter. Den detaljerade resvägen sker i fallande ordning, resan börjar längst upp och slutar längst ner. Se figur 18. I Appendix 9.1 finns alla skärmdumpar på Flutter-prototypen.



Figur 18. Bild på “Detaljerad resväg” sidan i Flutter-prototypen.

5.4 Utveckling med Uno Platform

Utvecklingen av Uno Platform-prototypen skedde efter det att utvecklingen av Flutter-prototypen var i stort sett klar. Funktionerna som implementerades för Uno Platform-prototypen är ekvivalenta med Flutter-prototypens funktioner. Däremot uppstod det några mindre avvikelser mellan prototyperna. Gränssnittet i Uno Platform-prototypen blev inte helt likt Flutters. Första avvikelsen befinners sig i filtreringsmenyn där det ska vara möjligt att välja datum och tid, som ska kunna användas för att hitta resor för just tiden man valde. I figur 15, 16 och 17 kan man se hur detta ser ut i Flutter.



Figur 19. Bild på Söksidan, med filtreringsmenyn samt kalendern och tidväljaren.

I figur 19 ser man hur detta ser ut i Uno Platform-prototypen. Kalendern centreras inte i mitten av skärmen och tiden väljs inte genom en klocka. Tiden väljs istället genom en meny där man kan ändra på tiden genom att trycka upp och ned på pilar för att öka respektive minska tiden.

Under utvecklingen tänktes det först användas CalendarDatePicker- och TimePicker-komponenter. Men dessa komponenter hade inte stöd för WebAssembly i Uno Platform. Lösningen blev istället att använda en JavaScript-komponent, Flatpickr. Det är en lättviktig “datetime”-picker som är skriven i JavsScript och som inte beror på andra bibliotek [29]. Med hjälp av Uno.WASM.Boostrapper-paketet, vilket ingår i standardpaketen som tillkommer med Uno Platform, så är det möjligt att lägga till JavaScript- och CSS-filer i

Uno-Wasm-applikationer [30]. Detta gjorde implementeringen mer komplex och tidskrävande än om det hade varit möjligt att använda vanliga komponenter.

En annan märkbar skillnad på gränssnittet är att texten i fälten “Från” och “Till” inte ändras till hållplatsen man valde. Det hittades ingen lösning på hur man sätter texten i fältet till namnet på hållplatsen man valde. När applikationen kördes i UWP istället för Wasm, fungerade det som tänkt, likt Flutter vilket man kan se i figur 14. På grund att det fungerar i UWP, kan man anta att det är en Wasm-specifik bugg. Detta löstes genom att skriva namnen på hållplatserna man valde under respektive fält, se figur 21.

För att implementera Google maps ska det egentligen fungera att använda MapControl-komponenten, men det finns inte stöd för WebAssembly.

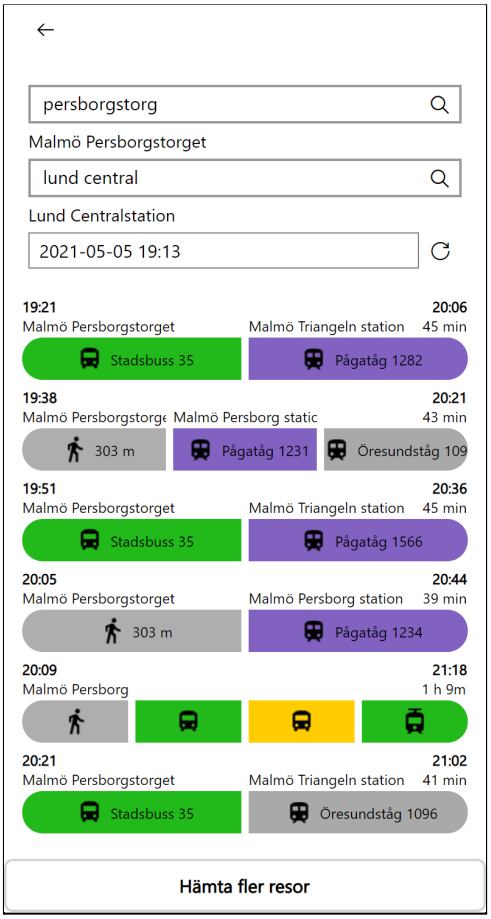
Feature	Android	iOS	Wasm
Display a map	X	X	
Display a path		X	
Customize pushpins with icon			
Fully template pushpins			
Show user's location on map			
Toggle native Locate-Me button visibility			
Toggle native Compass button visibility			

Figur 20. Tabell på vilka MapControl funktioner plattformar har stöd för.

Som man kan se i figur 20 finns det inget stöd alls för Wasm. Återigen blev lösningen att använda JavaScript. Detta blev mycket mer tidskrävande än det normalt hade varit om det fanns stöd för MapControl. För att hämta alla kartnålar, där varje kartnål representerar en hållplats, och sätta ut dem på kartan tänktes hållplatsinformationen läsas in från en textfil. Detta görs genom asynkron kod, men det upptäcktes att det inte finns ett enkelt sätt att anropa C# kod från JavsScript på ett asynkront sätt i Uno. Istället skapades en lista som innehåller alla hållplatser direkt, utan att ladda de från en textfil.

En sista avvikelse var hur gamla och nya resor hämtas. I Flutter gjordes detta genom att dra ändarna av listan. Befinner användaren sig i slutet av listan och drar uppåt hämtas nya resor, befinner man sig i toppen av listan och drar neråt hämtas gamla resor. I Uno hämtades nya och gamla resor automatiskt när man scrollade

till ändarna av listan. Men när det först görs en ny resesökning dyker det upp en knapp som kan användas för att hämta nya resor. Så fort användaren scrollar i listan försvinner knappen. Detta implementerades på grund av att man inte kunde hämta fler resor om det inte hämtades tillräckligt många resor för att listan ska vara scrollbar vid en ny sökning, Knappen behövdes inte i Flutter då man drar på listan för att hämta flera resor. I figur 21 kan man se hur knappen ser ut. I Appendix 9.2 finns alla skärmdumpar på Uno-prototypen.



Figur 21. Bild på Söksidan i Uno Platform-prototypen.

5.5 Djupgående jämförelse

Efter att utvecklingen av de två prototyperna är klar ska en jämförelse mellan Flutter och Uno Platform göras. Ett antal olika kriterier kommer att tas upp, dessa är: programmeringsspråk, kod, storlek, kompileringstid, tid att hämta data från API, prestanda, hot reload, paket och tillgängligheten av information. Genom att jämföra ramverken utifrån dessa kriterier ska det dras en slutsats om vilket ramverk som är bättre.

5.5.1 Programmeringsspråk

Koden i Flutter skrivs i programmeringsspråket Dart. Det är ett relativt nytt programmeringsspråk som kom ut 10:e oktober, 2011. Både Dart och Flutter utvecklas av Google. Syntaxen liknar Java/C. Även om det inte är ett av de mest populära språken bör det vara lätt för utvecklare att utveckla med Dart. All Dart specifik information som behövdes hittades på nätet utan problem. I Stack Overflow finns det 50,218 frågor angående Dart [31].

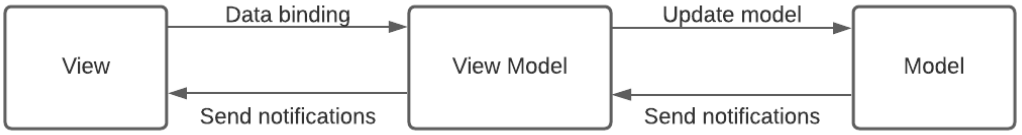
I Uno Platform används C#, vilket kom ut 2000. Det är ett väldigt känt språk som används av många utvecklare. Även JavaScript användes, men endast då det behövdes användas komponenter som Uno inte hade stöd för. Det förekom inte några problem att hitta information om C#. I Stack Overflow finns det 1,477,518 frågor angående C# [32].

Men eftersom C# är ett mycket använt språk som många utvecklare har erfarenhet, har Uno Platform en fördel när det gäller valet av programmeringsspråk.

5.5.2 Kod

Ytterligare en sak som är värt att nämna är skillnaden mellan strukturen på koden i de två olika prototyperna som utvecklades.

I Uno Platform användes ett designmönster som kallas för Model-View-ViewModel (MVVM), se figur 22. Poängen med mönstret är att separera på koden för gränssnittet, koden för datan och affärslogiken. Angående Flutter-prototypen användes inte ett designmönster på liknande sätt. Däremot delades koden fortfarande in i flera delar: sidor, datamodeller, widgets och verktyg,



Figur 22. MVVM-designmönster

Att använda MVVM-mönstret kändes naturligt i Uno Platform. Eftersom XAML används är gränssnittet redan separerat. Data binding användes för att hämta egenskaperna från view model som skulle visas i gränssnittet. Utvecklarna hade

inte mycket erfarenhet av att använda just MVVM-mönstret, men mönstret försöktes följas så gott som möjligt.

När det gäller Flutter fanns det inte en separation mellan gränssnittet och resten av koden. Det är möjligt att använda MVVM-mönstret i Flutter, men det kändes inte nödvändigt när prototypen utvecklades.

Något annat att ta hänsyn till är att Flutter-prototypen utvecklades först. Vilket innebär att när det började utvecklas med Uno Platform hade det redan bestämts hur hela prototypen skulle se ut och fungera. Detta underlättade utvecklingen och möjligen ledde till kod av högre kvalitet i Uno Platform. Hade Uno Platform-prototypen utvecklats först är det möjligt att MVVM-mönstret även hade använts i Flutter. Men återigen kändes det mer naturligt att använda mönstret i Uno.

Om MVVM-mönstret eller något liknande mönster hade använts för Flutter-prototypen hade nog kodkvalitén i Flutter-prototypen förbättrats. Eftersom det inte följdes något specifikt mönster i Flutter där gränssnittet separeras så kan man möjligen säga att Uno-prototypen ligger bättre till kodmässigt. Däremot är det svårt att säga om någon av ramverken har en fördel när det gäller kod allmänt.

5.5.3 Storlek

Angående storleken på respektive projektmapp fanns det en skillnad. När respektive projektmapp först hämtas har Flutter en storlek på 147 MB och Uno Platform har en storlek på 120 MB. Detta gäller endast vid en helt ny hämtning av mappen. Under utvecklingen, då man testar och kompilerar applikationen flera gånger, var storleken mycket större. Flutter hade en storlek på 2,03 GB och Uno hade en storlek på 296 MB. Detta gällde endast storleken på projektmappen och inte själva applikationen.

När en användare kör Flutter-applikationen på webbläsaren i mobilen kan användaren välja att installera appen. Detta görs genom att gå in i inställningar och trycka på installera applikation. Då kan man köra applikationen direkt från mobilen istället för att manuellt behöva besöka hemsidan som är värd för applikationen. Applikationen tar endast 276 kB, tar lite plats eftersom det inte är en riktig applikation. Mobilen lagrar inte filerna som behövs för att köra applikationen. I Uno Platform finns inte alternativet att installera appen på webbläsaren i mobilen.

Gällande storleken på projektmappen har Uno Platform en fördel. Vid en ny hämtning av projektmappen var det inte någon större skillnad, men under utvecklingen tog Flutter nästan sju gånger mer utrymme. Däremot spelar storleken ofta inte stor betydelse då de flesta utvecklare har mer än tillräckligt med utrymme på datorn. Flutter har en fördel när det gäller att kunna installera appen på mobilen, vilket inte går med Uno.

5.5.4 Kompileringstid

För att mäta kompileringstiden kompilerades prototyperna tio gånger på samma dator. Försöken utfördes direkt efter varandra. Mätningen av kompileringstiderna skedde både när prototyperna kompilerades till “debug mode” och “release mode”. Debug mode används när man håller på att utveckla, medan release mode är slutliga versionen som används när applikationen är redo att släppas ut. Resultatet finns i figur 23.

Kompileringstid (s)	Uno debug mode	Uno release mode	Flutter debug mode	Flutter release mode
Försök 1	19,991	19,632	16,2	27,4
Försök 2	19,606	19,588	16,5	26,8
Försök 3	19,440	19,571	17,3	27,1
Försök 4	19,487	19,392	16,7	27,4
Försök 5	19,275	19,657	16,4	27,2
Försök 6	19,208	19,246	16,7	27,4
Försök 7	19,215	19,578	16,6	27,1
Försök 8	19,212	19,730	16,6	28,1
Försök 9	19,175	19,575	17,4	28,3
Försök 10	19,201	20,078	16,2	27,3
Genomsnitt	19,381	19,6047	16,66	27,41

Figur 23. Tabell på kompileringstiden för Flutter och Uno Platform

När det gäller Uno Platform, är kompileringstiden för debug mode och release mode, 19,381 sekunder respektive 19,6047 sekunder. Release mode tar i genomsnitt 0,2237 sekunder längre. Detta innebär att i Uno Platform finns det inte någon större skillnad på kompileringstiden mellan debug mode och release mode.

Något att anmärka däremot är att när man först startar Visual Studio 2019 så kommer första kompileringsförsöket att ta cirka 6 sekunder längre än vad resultaten i tabellen visar. Detta verkar bero på att när man kompilerar startas Uno.SourceGeneration. Host i bakgrunden. När man sedan utför andra kompileringar är Uno.SourceGeneration.Host redan på, vilket leder till att kompileringstiden minskar med cirka 6 sekunder.

I genomsnitt tar det 16,66 sekunder att kompilera Flutter prototypen i debug mode. Medan i release mode tar det i genomsnitt 27,41 sekunder, en ökning på 10,75 sekunder från debug mode. Detta innebär att i Flutter finns det en relativt stor skillnad på kompileringstiden mellan debug mode och release mode.

Jämför man kompileringstiden mellan Uno Platform och Flutter så är resultatet att i debug mode tar Uno Platform i genomsnitt 2,721 sekunder längre. I release mode tar Flutter i genomsnitt 7,8053 sekunder längre.

Anledningen till den märkbara skillnaden i kompileringstid mellan Flutters debug och release mode beror nog på att när man kompilerar till release används en annan kompilator, dart2js. Medan i debug mode används dartdevc kompilatorn.

Sammanfattningsvis, när det gäller kompileringstiden har Flutter en fördel när det gäller debug mode. När det gäller release mode har Uno Platform en fördel. Däremot används för det mesta kompilering i debug mode under utvecklingen.

5.5.5 Tid att hämta data från API

Något som är intressant att undersöka är hur länge det tar för respektive ramverk att hämta data från ett API. Prototyperna hämtar datan som krävs för att visa resorna i listan, se figur 14, genom ett API-anrop till ResRobot - Reseplaneraren från hemsidan TrafikLab. Datan hämtas i JSON format.

Det gjordes tio försök per ramverk, mätningen av tid startade precis innan API-anropet och slutade efter att JSON datan har hämtats. I figur 24 och 25 ser man koden i Flutter respektive Uno Platform som användes för att mäta tiden på hur länge det tar att hämta data från ett API-anrop.

```dart
final stopwatch = Stopwatch()..start();

final responseData = await http.get(url).catchError((e) {});

print('Elapse: ${stopwatch.elapsed}');
```

Figur 24. Flutter kod på var mätningen av API anrop sker

```
Stopwatch sw = new Stopwatch();

sw.Start();

HttpResponseMessage response = await httpClient.GetAsync(requestUri);

sw.Stop();

Console.WriteLine("Elapsed={0}", sw.Elapsed);
```

Figur 25. Uno Platform kod på var mätningen av API anrop sker

Då mängden data varierar beroende på hållplatserna, tid och datum så bestämdes det att de olika försöken skulle ske under samma förutsättningar. Det valdes en resa mellan hållplatser som ligger långt från varandra och har därav många byten. Resan som valdes är mellan Malmö Triangeln och Kiruna Busstation, 2021-04-28 12:00. Anledningen till att denna resan valdes var för att det ansågs att undersökningen kommer att få ett bättre resultat om det hämtas större mängder data. Resultatet finns i figur 26.

Tid att hämta data från API (s)	Uno Platform	Flutter
Försök 1	4,4493050	4,200495
Försök 2	4,3058300	3,970955
Försök 3	4,2662750	4,062265
Försök 4	4,2969750	4,050475
Försök 5	4,3742000	4,023125
Försök 6	4,1622650	4,614490
Försök 7	4,4444850	4,136855
Försök 8	4,2180300	4,447419
Försök 9	4,4999500	4,755340
Försök 10	4,1496450	4,273224
Genomsnitt	4,3166960	4,253463

Figur 26. Tabell på mätningen av tid för att hämta data från API

I genomsnitt tog det Uno Platform 4,316696 sekunder att hämta datan från API:t medan det tog 4,253463 sekunder för Flutter. Flutter var i genomsnitt 0,063233 sekunder snabbare än Uno Platform gällande hämtning av data från API. Detta är en så pass liten skillnad att man kan dra slutsatsen att ramverken hämtar data från ett API lika snabbt. Däremot verkar Flutter ha en större variation på tiden det tar, från 3,970955 till 4,755340 sekunder medan det tar mellan 4,1496450 och 4,4999500 sekunder i Uno Platform.

Sedan undersöktes det hur lång tid det tar för användargränssnittet i Uno att uppdateras med listan av resor som precis hämtats. I Flutter var det ingen märkbar skillnad mellan att datan hämtades från API:t och att gränssnittet uppdaterades. Genom att använda Chrome DevTools Console kunde tiden mellan att datan har hämtas från API:t till att listan dyker upp i gränssnittet mätas. Det gjordes fem försök, resultatet finns i figur 27.

Tid (s)	3,754	3,14	3,39	3,391	3,442

Figur 27. Tabell på mätningen av tid för gränssnittet att uppdateras i Uno.

Detta resulterar i ett genomsnitt på 3,4234 sekunder.

När det gäller tiden det tar att göra ett anrop till ett API och hämta data, finns det inte någon större skillnad mellan Flutter och Uno Platform. Däremot finns det en större skillnad på tiden från att datan har hämtats från API:t till att listan med resor visas i gränssnittet. I Flutter sker detta omedelbart utan någon märkbar dröjning, medan det tar i genomsnitt 3,4234 sekunder för Uno Platform att rendera listan. Det är svårt att hitta orsaken till denna dröjningen, om det är själva implementeringen eller Uno-Wasm.

5.5.6 Prestanda

En av de viktigaste kriterierna är prestanda. Tidigt under utvecklingen kunde det konstateras att Flutter hade bättre prestanda. Med detta menas bland annat att övergången mellan sidor och renderingen av gränssnittskomponenter var snabbare. Skillnaden på hur snabbt renderingen sker kan man se i kapitel 5.5.5.

Efter att ha spenderat ungefär en månad med varje ramverk och prövat på många av de två olika ramverkens funktioner kunde det konstateras att Flutter kändes överlägsen.

Prestanda av applikationer kan även påverkas av själva implementeringen av kod. Däremot ansågs det att detta var inte anledningen till skillnaden i prestanda. När applikationen kördes i UWP iställer för Wasm i Uno Platform skedde övergångarna mellan sidorna snabbare, trots att koden var samma. Detta, i kombination med andra brister som upptäcktes för Wasm, gör att anledningen till prestandan tros inte vara koden. Jämfört med Flutter ansågs Uno vara ett omoget ramverk, ramverkets utvecklare fortsätter att förbättra prestandan men jämfört med Flutter var prestandan sämre.

5.5.7 Hot reload

Vidare märktes det en skillnad mellan ramverken inom det som kallas för “hot reloading” som innebär att en körande applikation uppdateras vid ändring av kod. Det vill säga att det inte behövs att man kompilerar om och startar om applikationen för att funktioner inom körande applikationen ska uppdateras.

Utav de två ramverken fungerade hot reloading som tänkt endast för Flutter. Angående Uno Platform ska det fungera med hot reloading när man använder sig av Uno-Wasm men det fungerade inte riktigt. När det användes JavaScript-komponenter eller komponenter från andra paket utöver de vanliga

som tillkommer med Uno Platform slutade hot reload fungera. Det lyckades inte hitta en lösning till detta. Körde man med UWP i Uno så fungerade hot reload även när man använde komponenter från andra paket, däremot har inte UWP stöd för JavaScript-komponenter.

Detta medförde att vid varje ändring av koden var det nödvändigt att kompilera och starta om applikationen. På så sätt blev det tidskrävande att testa applikationen jämfört med Flutter vars stöd för hot reloading fungerade felfritt. Utifrån detta har Flutter en stor fördel när det gäller hot reload.

5.5.8 Paket

Något som skiljde sig mycket mellan Flutter och Uno Platform när de två prototyperna utvecklades är antalet paket som användes. Vid utvecklingen av Flutter-prototypen fanns det ett stort utbud av paket som användes för att underlätta utvecklingen. Det gjorde att det sparades en hel del tid. Utbudet av paket som Flutter erbjöd hade däremot inte Uno Platform.

Vid utvecklingen av Uno Platform-prototypen användes endast två paket utöver standardpaketen som kom med installationen av Visual Studio och Uno. Paketen som användes var Newtonsoft.Json som användes för att hantera JSON datan från API:t samt Uno.Microsoft.Toolkit som användes för utseendet av listan.

I Flutter hämtades det flera olika paket som geolocator, date_time_picker, pull_to_refresh och google_maps_flutter_web. Paketen som används i Flutter hämtades från pub.dev som har över 15926 Flutter-paket [33]. Paket för Uno Platform hämtades från nuget.org, det finns endast 82 Uno Platform-paket [34]. Bland de 82 paketen är många standardpaketen som följer med installeringen av ramverket.

Flutter har en fördel när det gäller utbudet av paket, användandet av paket underlättar utvecklingen. Som utvecklare har man större möjligheter med ett stort utbud av paket.

5.5.9 Tillgängligheten av information

När det kommer till tillgängligheten av information för de två olika ramverken finns det en stor skillnad mellan Flutter och Uno Platform. Det fanns mycket information och exempel ute på nätet för Flutter. Det underlättade mycket under utvecklingen av Flutter-prototypen.

Angående Uno-ramverket var det oftast svårt att hitta informationen man efterfrågade. Det fanns väldigt lite information att hämta från nätet. Ibland fanns det inte någon information alls. Största delen av information hämtades genom att söka efter UWP-specifik information. Men vid flera tillfällen fungerade saker i UWP men inte i WebAssembly.

I Stack Overflow finns det endast 246 frågor angående Uno Platform medan det finns 86,624 frågor om Flutter. Uno har 4,800 stjärnor i GitHub medan Flutter har 120,000 stjärnor.

Flutter har en stor fördel när det gäller populariteten och tillgängligheten av information. Detta underlättar utvecklingen enormt då sannolikheten är stor att det finns information om det man efterfrågar.

6 Slutsats

Två prototyper utvecklades, en med hjälp av Flutter-ramverket och den andra med hjälp av Uno-ramverket. Både prototyperna utvecklades som webbapplikationer. Prototyperna utvecklades med samma funktioner och nästan likadana användargränssnitt men med hjälp av två olika programmeringsspråk. Programmeringsspråken som användes för att utveckla de två prototyperna var Dart och C#.

För att utveckla Uno-prototypen användes MVVM-mönstret. Under utvecklingen av Flutter-prototypen användes däremot inget mönster. Storleken på de två slutgiltiga prototyperna varierade. När man först importerade projektet för vardera prototyp var de ungefär lika stora. När man däremot byggde och körde prototyperna flera gånger blev det en stor skillnad mellan storleken. Då blev Uno-prototypen runt sju gånger större än Flutter-prototypen. Angående kompileringstiden för de två prototyperna så varierade det. Vid kompilering av prototyperna där det användes debug mode kompilerades Flutter-prototypen 2,721 sekunder snabbare än Uno-prototypen. Under kompilering av release mode kompilerades Uno-prototypen 7,8053 sekunder snabbare. Vid API-anrop var Flutter-prototypen i genomsnitt 0,063233 sekunder snabbare än Uno-prototypen. När det gällde hur långt tid det tog att rendera listan med resor så tog det i genomsnitt 3,4234 sekunder för en lång resa i Uno Platform, renderingen skedde omedelbart i Flutter. Detta är en stor fördel för Flutter.

Angående de två prototypernas prestanda så presterade Flutter-prototypen bättre. Flutter-prototypen hade även fördelar inom hot reload som inte Uno-prototypen har. Hot reload fungerade som det ska för Flutter-prototypen. I fråga om Uno-prototypen fungerade hot reload men inte när man använde sig av JavaScript-komponenter eller komponenter från andra paket utöver de vanliga. När det gäller antalet paket som fanns tillgängliga att använda under utvecklingen så var det en stor skillnad. Paketen som användes i Flutter-protoypen hämtades från pub.dev som har över 15927 Flutter-paket. Paket som användes under utvecklingen av Uno-prototypen hämtades från nuget.org, där finns endast 82 Uno Platform-paket. Paketen spelade en stor roll under utveckling av prototyperna och där hade Flutter-prototypen fördelen med att det fanns fler paket att använda sig av. Sist men inte minst var även Flutter överlägsen angående hur mycket information som finns att hitta på nätet.

Utifrån allt detta drogs slutsatsen att Flutter är ett bättre ramverk än Uno Platform.

Under examensarbetets gång har alla problemformuleringar lösts och besvarats.

Vilka kriterier är lämpliga att ha för den initiala jämförelsen?

- Vilka möjligheter finns det för att få hjälp på internet?
  - Hur populärt är ramverket?

Detta kriteriet valdes eftersom när man utvecklar så underlättar det om man kan hitta information på nätet. Om det är svårt att hitta information är utvecklingen mer tidskrävande. Möjligheten att få hjälp på internet har en stor korrelation med hur populärt ramverket är. Det finns en anledning till varför vissa ramverk är populära och andra inte är. Om ett ramverk är gammalt och har en låg popularitet är ramverket inte intressant för just detta examensarbetet.

- När introducerades ramverket?

Kriteriet ansågs lämpligt eftersom att årtalet när ett ramverk introducerades spelar stor roll för bland annat hur möjligheterna att få hjälp på internet är. Ett nyare ramverk har mindre information på nätet vilket kan göra det svårt för en utvecklare som använder detta specifika ramverket. Ett gammal ramverk har förmodligen fler frågor och lösningar vilket underlättar för utvecklare som försöker hitta lösningar på problem som de har. Vidare spelar årtalet roll när det gäller hur modern ramverket är. Ett gammalt ramverk kanske inte har de senaste komponenterna och verktygen som kanske nyare ramverk som introducerades senare har. Däremot har nya ramverk oftast fler buggar än äldre ramverk.

- Kan man utveckla webbapplikationer som automatiskt anpassar sig efter vilket operativsystem som enheten körs på utan att använda nativelösningar?

Eftersom att syftet med detta examensarbete är att utvärdera två eller flera olika cross-platform-ramverk så kändes det naturligt att ha med detta kriteriet.

- Är det gratis att använda ramverket?

Ramverk som är helt gratis föredras. Det är mindre komplicerat att ladda ner ramverk utan att behöva köpa de. Ramverken kan testas av vem som helst. Dessutom finns det många ramverk som är gratis, därför ansågs det av examensarbetarna onödigt att behöva köpa ett ramverk.

Vilka kriterier är lämpliga att ha vid den djupgående jämförelsen mellan ramverken som valdes ut i den initiala jämförelsen och Uno Platform?

Kriterierna som ansågs vara lämpliga är följande:

- Programmeringsspråk
- Kod
- Storlek
- Kompileringstid
- Tid att hämta data från API,
- Prestanda
- Hot reload
- Paket
- Tillgänglighet av information.

Det som ansågs vara viktigast när det gäller om ett ramverk är bra eller inte är själva prestandan men även upplevelsen för utvecklarna. Dessa kriterier består av både delarna, hur ramverket presterar: storlek, kompileringstid, tid att hämta data från API och prestanda. Men även hur det är att utveckla med ramverket: programmeringsspråk, kod, hot reload, paket och tillgänglighet av information.

Vilken typ av applikation är lämplig som prototyp?

- **Vilken funktionalitet ska finnas med?**

Eftersom att examensarbetet görs tillsammans med Softhouse som har ett samarbete med Skånetrafiken så var en reseplanerare ett intressant val av typ av applikation. Funktionaliteten som skulle finnas var lätt att bestämma då de flesta reseplanerare har liknande funktioner men eftersom inte mycket tid fanns begränsades antalet funktioner som skulle implementeras. Funktionerna som valdes att implementeras valdes med hjälp av handledaren från Softhouse. Funktionerna som valdes är att kunna göra en resesökning där man har start- och slutdestination där man även har möjligheten att använda sig av en kartfunktion för att göra en sökning. Vidare valdes funktionen att kunna filtrera en sökning efter datum och tid.

Hur resultatet bidrar till att uppfylla syftet

Syftet med detta examensarbete är att utvärdera två eller flera olika cross-platform-ramverk vilket också görs i resultatet. Resultatet jämför de två olika ramverkens programmeringsspråk. Sedan tar resultatet också upp kodstrukturen som finns inom de två olika ramverken. Vidare tas det upp om

skillnaderna mellan storleken på projekten som de två prototyperna består av. Ytterligare diskuteras kompileringstiden som de två prototyperna har. Därtill kommer också en diskussion om prestandan för de två prototyperna samt om antalet paket som de två ramverken erbjuder. Sist men inte minst lyfts ämnet om tillgängligheten av information upp där det diskuteras om hur lätt det är att hitta information om respektive ramverk. Det som diskuteras i resultatet kompletterar mer eller mindre syftet.

Hur resultatet kommer att användas

Softhouse kommer ha nytta av resultatet på det sättet att de kommer att kunna dra slutsats om någon av dessa cross-platform-ramverk är ramverk som företaget kan tänkas använda för att utveckla deras framtida applikationer.

6.1 Reflektioner över etiska aspekter

De två prototyperna som utvecklades är webbapplikationer som kan användas av samhället. Prototyperna använder sig av data som hämtas från liknande API:er som även Skånetrafiken använder sig av. Därför kan människor som använder prototyperna vara rätt bekväma med att tiderna som prototyperna visar stämmer. En stor samhällsnytta som dessa prototyper tillför är att resesökningen i prototyperna fungerar inom hela Sverige. Det gör det möjligt att alla människor runt om i Sverige kan använda sig av prototyperna.

6.2 Framtida utvecklingsmöjligheter

Det finns flera utvecklingsmöjligheter som hade varit intressanta. Något som kan tänkas utvecklas inom prototyperna är att implementera ett biljettsystem där man kan köpa biljett/biljetter för resorna man söker. Då hade prototyperna varit helt kompletta. En annan sak som kan förbättras är att implementera felhantering inom koden på fler ställen än vad det är just nu. Med felhanteringen får användaren eller utvecklaren felmeddelanden när något fel inträffar i applikationen som körs.

Vidare kan Uno-prototypen förbättras lite när det gäller användargränssnittet. Uno-prototypens liknar inte Flutter-prototypen till fullo. Möjligen hitta lösningar till skillnaderna som beskrivs i kapitel 5.4. En ytterligare funktion som kan tänkas förbättras är strukturen på koden inom Flutter-prototypen genom att använda något arkitekturmönster. Just nu är koden i Flutter-prototypen inte uppdelad i någon struktur där man exempelvis skiljer koden för vyn och modellen som i Uno-prototypen. En ytterligare sak som kan tänkas förbättras för både

prototyperna är att utveckla prototyperna så pass mycket att de ska efterlikna alla Skånetrafikens funktioner.

7 Terminologi

API: Förkortning för “Application Program Interface” och är ett verktyg som används för att program ska kunna kommunicera med varandra samt utbyta information.

Cross-platform: Ett program som kan köras på flera operativsystem.

Kodbas: Alla kodfiler som en applikation använder sig av för att kunna fungera.

Nativeapplikation: En applikation som är utvecklad för ett bestämt operativsystem som exempelvis Android eller iOS. Den är inte utvecklad för att köras i en webbläsare.

Renderingsmotor: Ett program som använder sig av märkspråk som indata för att sedan visa upp det på skärmen som utdata.

UWP: Universal Windows Platform är en plattform som används för att skapa applikationer som är specifikt för Windows.

WebAssembly: Ett programmeringsspråk där koden representeras i binär form vilket webbläsare har förmågan att förstå och rendera.

Webbapplikation: En applikation som körs med hjälp av en webbläsare.

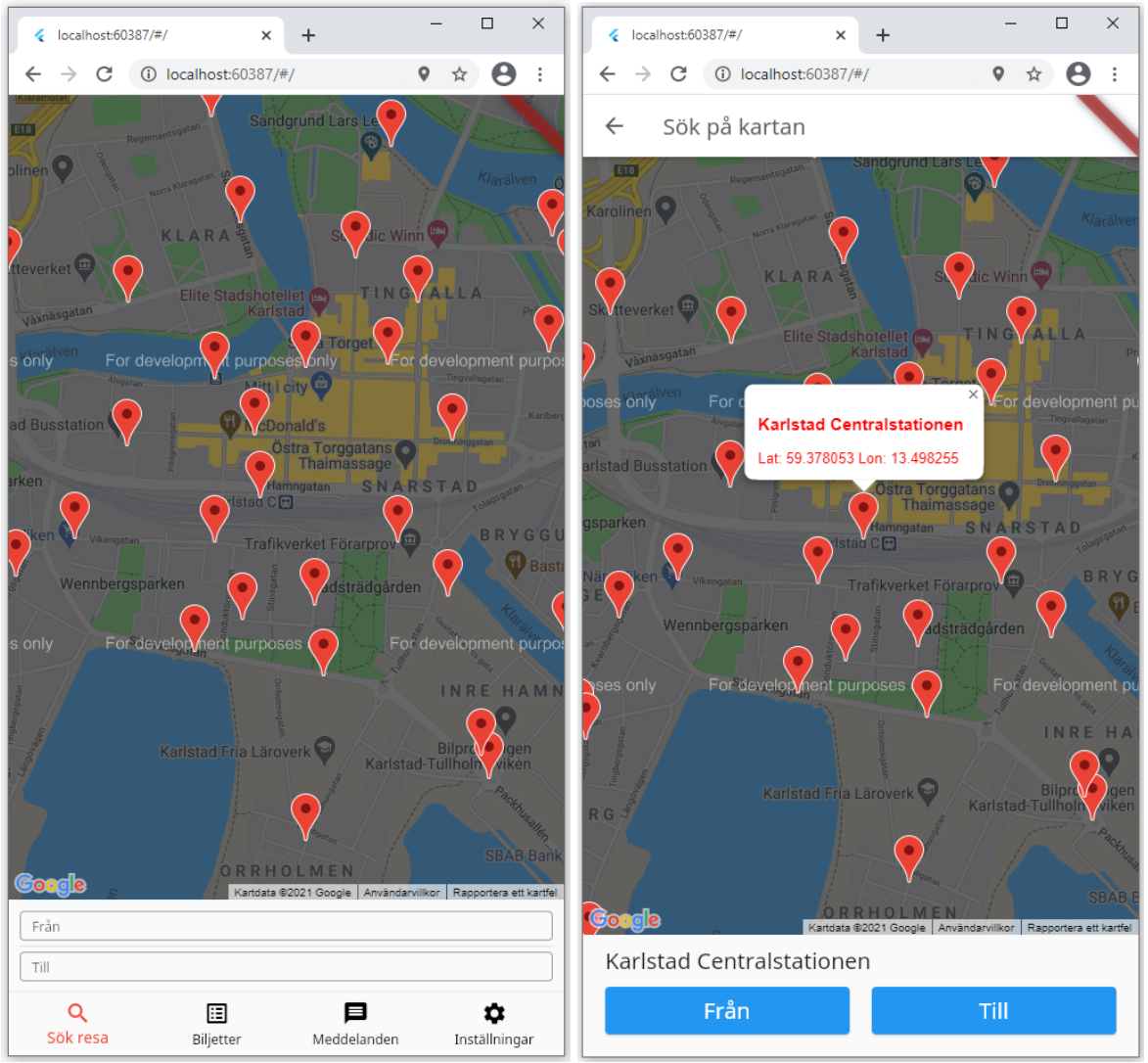
8 Källförteckning

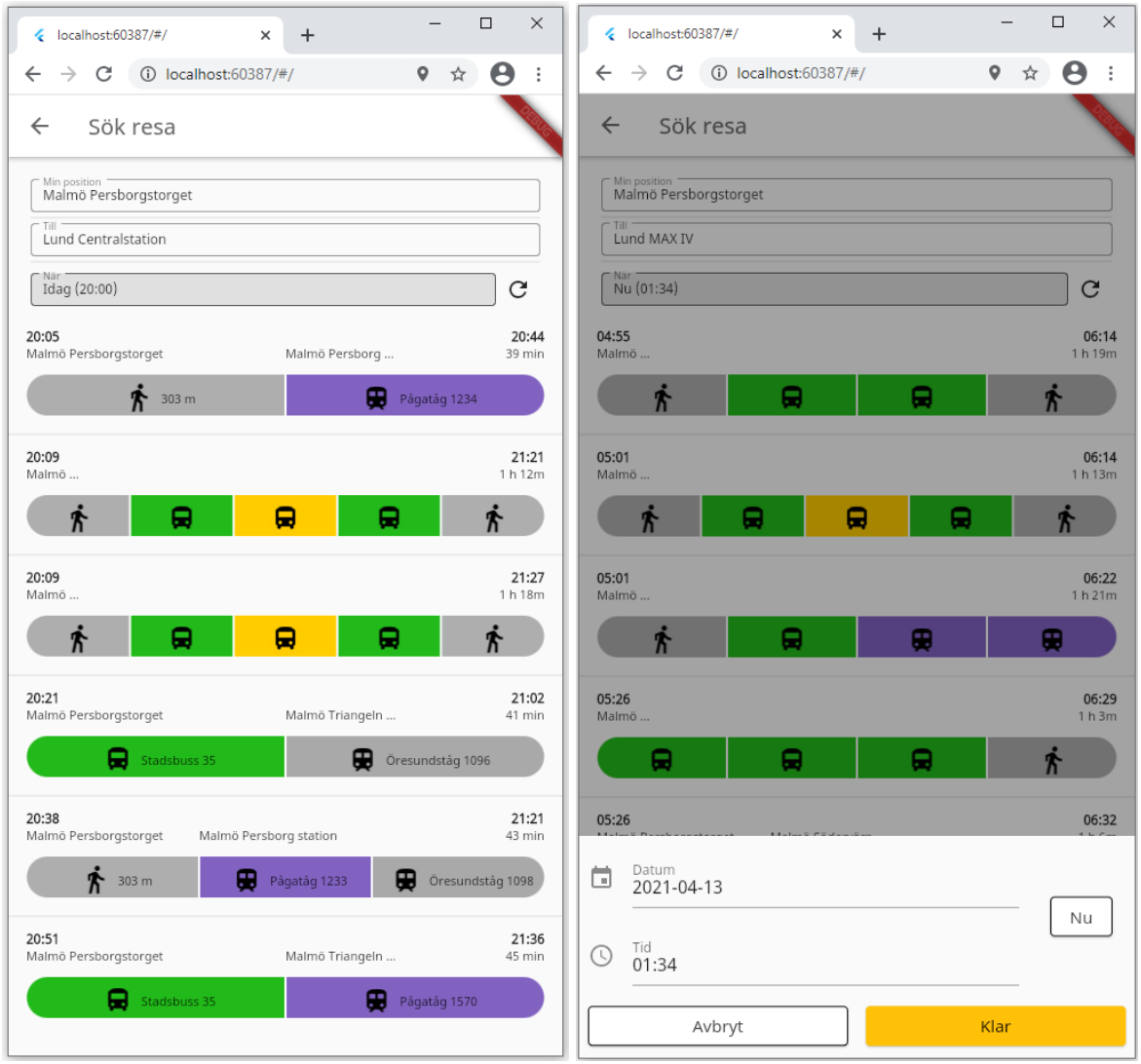
[1] Softhouse. Företagets officiella hemsida. https://www.softhouse.se/ (Hämtad 2021-01-05)
[2] uptech. Native vs Cross-platform. https://uptech.team/blog/native-vs-cross-platform-app-development (Hämtad 2021-01-10)
[3] Uno Platform. Ramverkets officiella hemsida. https://platform.uno/ (Hämtad 2021-01-10)
[4] Uno Platform. Officiell dokumentation. https://platform.uno/docs/articles/intro.html (Hämtad 2021-02-20)
[5] Uno Platform. Hur det fungerar. https://platform.uno/how-it-works/ (Hämtad 2021-05-10)
[6] Flutter. SDK utsläpp. https://flutter.dev/docs/development/tools/sdk/releases (Hämtad 2021-02-20)
[7] Flutter. Flutters build modes. https://flutter.dev/docs/testing/build-modes (Hämtad 2021-05-06)
[8] Flutter. Renderingsmotor för webben. https://flutter.dev/docs/development/tools/web-renderers (Hämtad 2021-02-08)
[9] Flutter. Introduktion till widgets. https://flutter.dev/docs/development/ui/widgets-intro (Hämtad 2021-02-08)
[10] Softhouse. Scrum på 5 minuter. https://www.softhouse.se/wp-content/uploads/In5_Scrum_se.pdf (Hämtad 2021-03-01)
[11] Trello. Trello framsida. https://trello.com/en (Hämtad 2021-05-08)
[12] GitHub Pages. GitHub Pages framsida. https://pages.github.com/ (Hämtad 2021-05-08)
[13] GitHub. Flutters GitHub sida. https://github.com/flutter/flutter (Hämtad 2021-05-11)
[14] Stack Overflow. Frågor med taggen [flutter]. https://stackoverflow.com/questions/tagged/flutter (Hämtad 2021-05-11)
[15] Stack Overflow. 2020 enkät för utvecklare. https://insights.stackoverflow.com/survey/2020 (Hämtad 2021-02-20)
[16] Flutter Samples. Lista med Flutter appar. https://flutter.github.io/samples/# (Hämtad 2021-02-20)
[17] GitHub. React releases. https://github.com/facebook/react/releases?after=v0.8.0 (Hämtad 2021-05-11)
[18] C#Corner. Varför och vad är React.js. https://www.c-sharpcorner.com/article/what-and-why-reactjs/ (Hämtad 2021-02-20)

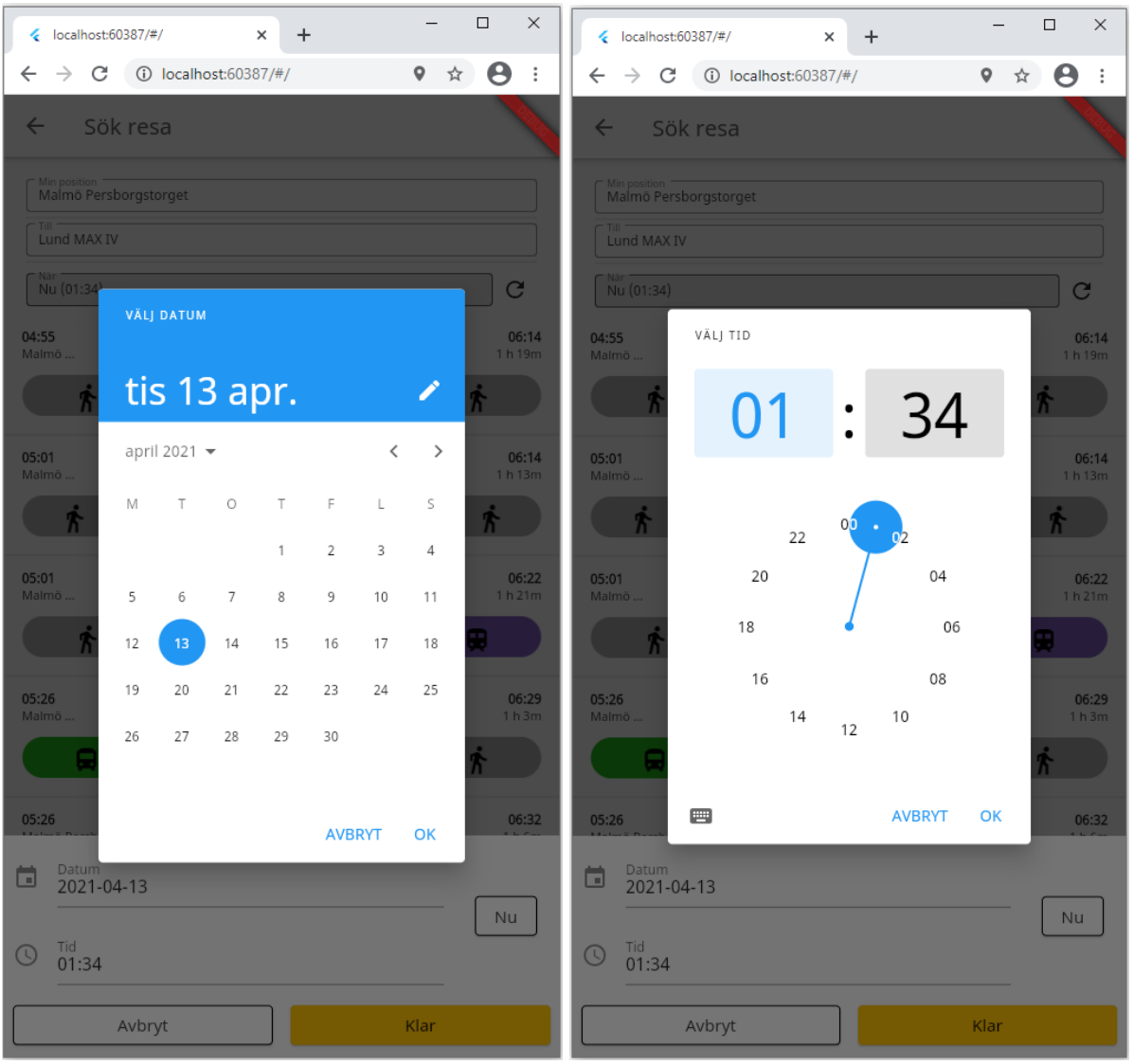
[19] Stack Overflow. Frågor med taggen [reactjs]. https://stackoverflow.com/questions/tagged/reactjs (Hämtad 2021-05-11).
[20] GitHub. Reacts GitHub sida. https://github.com/facebook/react (Hämtad 2021-05-11)
[21] React.js. Exempel på projekt utvecklade med React. https://reactjs.org/community/examples.html (Hämtad 2021-05-10)
[22] JavaTPoint. Ionic Frameworks historia. https://www.javatpoint.com/ionic-history (Hämtad 2021-02-20)
[23] Ionic Framework. Ionic Frameworks officiella dokumentation. https://ionicframework.com/docs (Hämtad 2021-02-20)
[24] Ionic Framework. Progressive Web Apps. https://ionicframework.com/docs/core-concepts/what-are-progressive-web-apps (Hämtad 2021-02-20)
[25] GitHub. Ionic Frameworks GitHub sida. https://github.com/ionic-team/ionic-framework (Hämtad 2021-05-11)
[26] Stack Overflow. Frågor med taggen [ionic-framework]. https://stackoverflow.com/questions/tagged/ionic-framework (Hämtad 2021-05-11)
[27] Ionic Framework. Framsidan. https://ionicframework.com/ (Hämtad 2021-05-09)
[28] Star Track. Demo Ionic App. https://startrack-ng.web.app/browse (Hämtad 2021-02-20)
[29] Flatpickr. Dokumentation om Flatpickr komponenten. https://flatpickr.js.org/ (Hämtad 2021-04-27)
[30] Uno Platform. Inbäddning av JavaScript-komponenter i Uno-WASM. https://qa.website.platform.uno/docs/articles/interop/wasm-javascript-1.html (Hämtad 2021-04-27)
[31] Stack Overflow. Frågor med taggen [dart]. https://stackoverflow.com/questions/tagged/dart (Hämtad 2021-05-11)
[32] Stack Overflow. Frågor med taggen [c#]. https://stackoverflow.com/questions/tagged/c%23 (Hämtad 2021-05-11)
[33] pub.dev. Flutter paket. https://pub.dev/flutter/packages (Hämtad 2021-05-11)
[34] nuget. Uno Platform-paket. https://www.nuget.org/packages?q=uno+platform (Hämtad 2021-05-11)

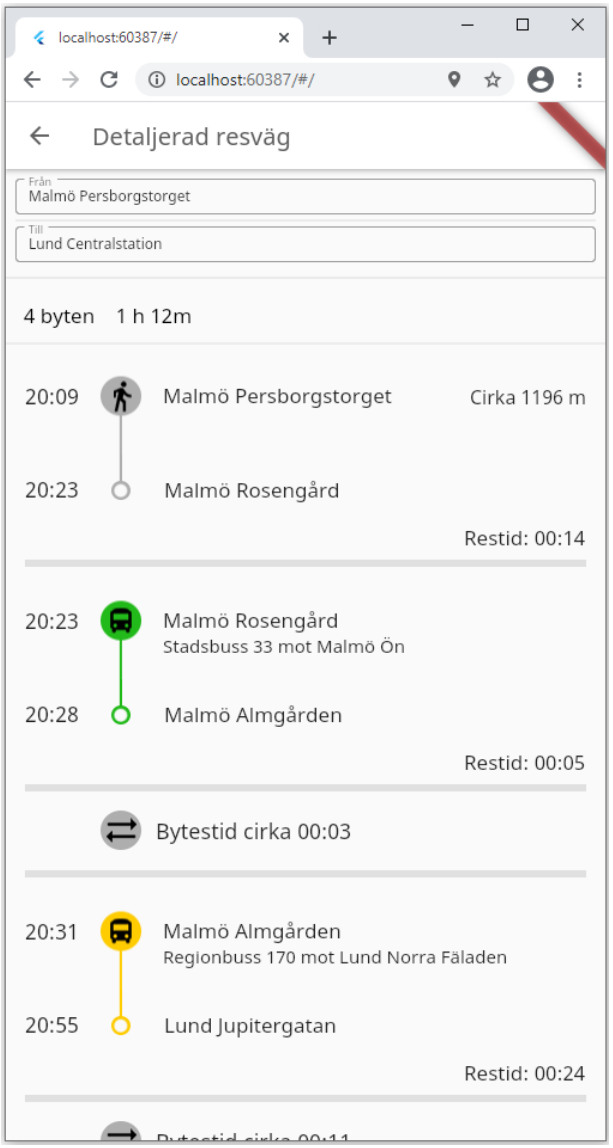
9 Appendix

9.1 Flutter skärmdumpar

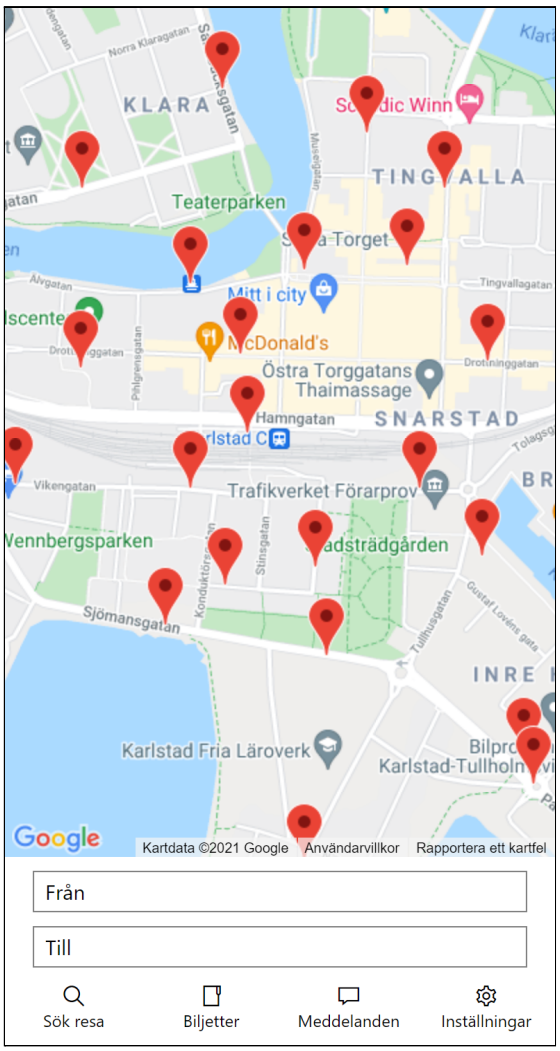


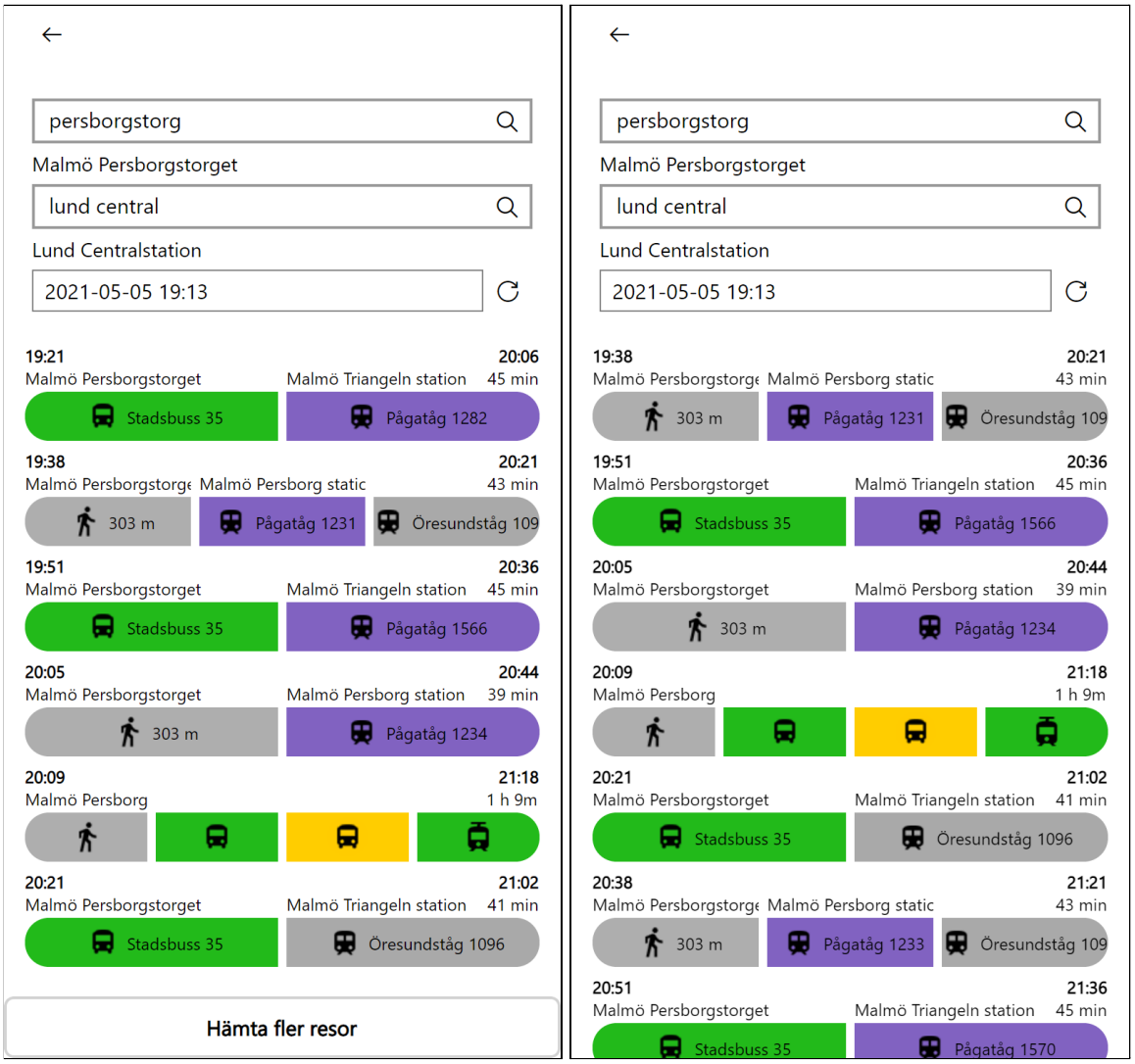

localhost:60387/#/
Sök resa
Min position
Malmö Persborgstorget
Till
Lund Centralstation
När
Idag (20:00)
20:05
Malmö Persborgstorget
20:44
Malmö Persborg ...
39 min
303 m
Pågatåg 1234
20:09
Malmö ...
21:21
1 h 12m
20:09
Malmö ...
21:27
1 h 18m
20:21
Malmö Persborgstorget
21:02
Malmö Triangeln ...
41 min
Stadsbuss 35
Öresundståg 1096
20:38
Malmö Persborgstorget
Malmö Persborg station
21:21
43 min
303 m
Pågatåg 1233
Öresundståg 1098
20:51
Malmö Persborgstorget
21:36
Malmö Triangeln ...
45 min
Stadsbuss 35
Pågatåg 1570
localhost:60387/#/
Sök resa
Min position
Malmö Persborgstorget
Till
Lund MAX IV
När
Nu (01:34)
04:55
Malmö ...
06:14
1 h 19m
05:01
Malmö ...
06:14
1 h 13m
05:01
Malmö ...
06:22
1 h 21m
05:26
Malmö ...
06:29
1 h 3m
05:26
06:32
Datum
2021-04-13
Nu
Tid
01:34
Avbryt
Klar

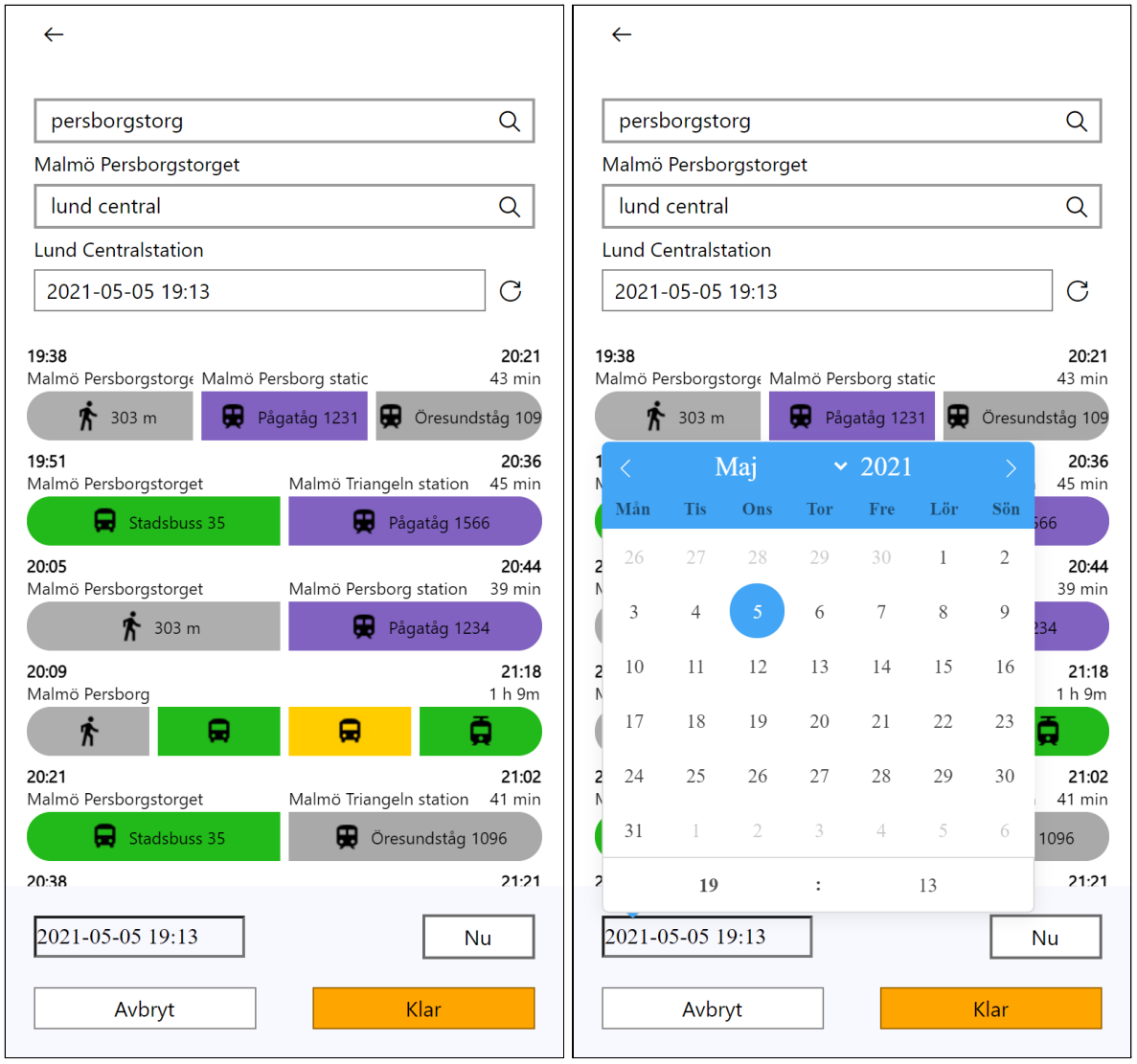

localhost:60387/#/
Sök resa
Min position
Malmö Persborgstorget
Till
Lund MAX IV
När
Nu (01:34)
VÄLJ DATUM
tis 13 apr.
april 2021
M T O T F L S
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30
AVBRYT
OK
Datum
2021-04-13
Nu
Tid
01:34
Avbryt
Klar
VÄLJ TID
01 : 34
00 02 04 06 08 10 12 14 16 18 20 22
AVBRYT
OK

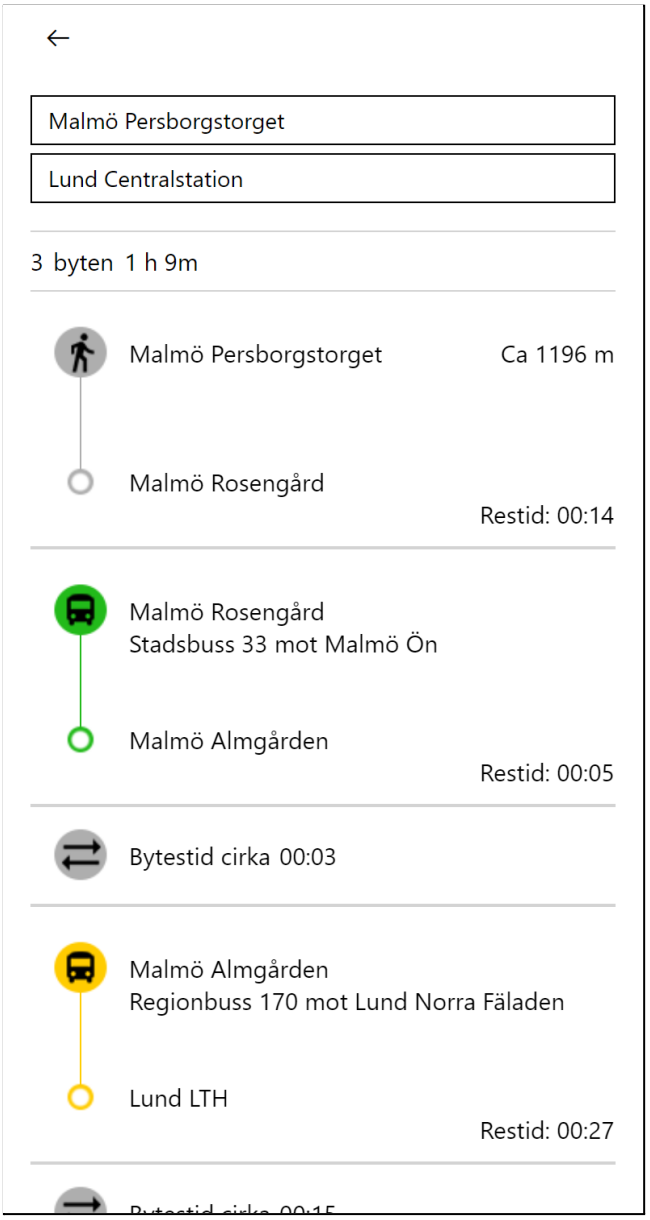

localhost:60387/#/
localhost:60387/#/
Detaljerad resväg
Från
Malmö Persborgstorget
Till
Lund Centralstation
4 byten 1 h 12m
20:09
Malmö Persborgstorget
Cirka 1196 m
20:23
Malmö Rosengård
Restid: 00:14
20:23
Malmö Rosengård
Stadsbuss 33 mot Malmö Ön
20:28
Malmö Almgården
Restid: 00:05
Bytestid cirka 00:03
20:31
Malmö Almgården
Regionbuss 170 mot Lund Norra Fäladen
20:55
Lund Jupitergatan
Restid: 00:24

9.2 Uno Platform skärmdumpar




persborgstorg
Malmö Persborgstorget
lund central
Lund Centralstation
2021-05-05 19:13
19:21
20:06
Malmö Persborgstorget
Malmö Triangeln station
45 min
Stadsbuss 35
Pågatåg 1282
19:38
20:21
Malmö Persborgstorge
Malmö Persborg static
43 min
303 m
Pågatåg 1231
Öresundståg 109
19:51
20:36
Malmö Persborgstorget
Malmö Triangeln station
45 min
Stadsbuss 35
Pågatåg 1566
20:05
20:44
Malmö Persborgstorget
Malmö Persborg station
39 min
303 m
Pågatåg 1234
20:09
21:18
Malmö Persborg
1 h 9m
20:21
21:02
Malmö Persborgstorget
Malmö Triangeln station
41 min
Stadsbuss 35
Öresundståg 1096
Hämta fler resor
persborgstorg
Malmö Persborgstorget
lund central
Lund Centralstation
2021-05-05 19:13
19:38
20:21
Malmö Persborgstorge
Malmö Persborg static
43 min
303 m
Pågatåg 1231
Öresundståg 109
19:51
20:36
Malmö Persborgstorget
Malmö Triangeln station
45 min
Stadsbuss 35
Pågatåg 1566
20:05
20:44
Malmö Persborgstorget
Malmö Persborg station
39 min
303 m
Pågatåg 1234
20:09
21:18
Malmö Persborg
1 h 9m
20:21
21:02
Malmö Persborgstorget
Malmö Triangeln station
41 min
Stadsbuss 35
Öresundståg 1096
20:38
21:21
Malmö Persborgstorge
Malmö Persborg static
43 min
303 m
Pågatåg 1233
Öresundståg 109
20:51
21:36
Malmö Persborgstorget
Malmö Triangeln station
45 min
Stadsbuss 35
Pågatåg 1570


persborgstorg
Malmö Persborgstorget
lund central
Lund Centralstation
2021-05-05 19:13
19:38
Malmö Persborgstorge
Malmö Persborg static
20:21
43 min
303 m
Pågatåg 1231
Öresundståg 109
19:51
Malmö Persborgstorget
Malmö Triangeln station
20:36
45 min
Stadsbuss 35
Pågatåg 1566
20:05
Malmö Persborgstorget
Malmö Persborg station
20:44
39 min
303 m
Pågatåg 1234
20:09
Malmö Persborg
21:18
1 h 9m
20:21
Malmö Persborgstorget
Malmö Triangeln station
21:02
41 min
Stadsbuss 35
Öresundståg 1096
20:38
21:21
2021-05-05 19:13
Nu
Avbryt
Klar
Maj
2021
Mån Tis Ons Tor Fre Lör Sön
26 27 28 29 30 1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31 1 2 3 4 5 6
19 : 13


Malmö Persborgstorget
Lund Centralstation
3 byten 1 h 9m
Malmö Persborgstorget
Ca 1196 m
Malmö Rosengård
Restid: 00:14
Malmö Rosengård
Stadsbuss 33 mot Malmö Ön
Malmö Almgården
Restid: 00:05
Bytestid cirka 00:03
Malmö Almgården
Regionbuss 170 mot Lund Norra Fäladen
Lund LTH
Restid: 00:27