computers

MDPI

Article

The Use of Geolocation to Manage Passenger Mobility between Airports and Cities †

Antonio Sarasa-Cabezuelo

Department of Computer Systems and Computing, School of Computer Science, Complutensian University of Madrid, 28040 Madrid, Spain; asarasa@ucm.es

† This paper is an extended version of a paper presented in 1st International Workshop on the Web of Things for Humans (WoT4H), Helsinki, Finland, 9 June 2020.

Received: 18 August 2020; Accepted: 3 September 2020; Published: 11 September 2020

Abstract: A general problem in large cities is mobility. Every day, there are incidents (accidents, construction, or meteorological events) that increase the duration of the journeys in a city and exert negative effects on the lives of citizens. A particular case of this situation is communications with airports. Shuttles are a type of private transport service that operates in airports. The number of passengers carried by shuttles is small. Moreover, shuttles transport passengers to their final destinations and their prices are more competitive than those other private services. However, shuttle services have a limitation with respect to their routes due to the many different destinations of passengers. For this reason, it is important to be able to calculate the best route to optimize the journey. This work presents an Android application that implements a value-added service to plan the route of a shuttle and manage its service (passengers, journeys, routes, etc.). This application combines information from different sources, such as geolocation information from the shuttle driver's mobile phone, information on the passengers, and results obtained from the service offered by MapBox to calculate the optimized route between several destinations.

Keywords: geolocation; optimization; App android; MapBox; sensors

1. Introduction

One of the main problems in large cities is the mobility of citizens [1]. In most cities, there are various types of public transport, such as buses, subways, trams, etc. However, many citizens prefer to use their own vehicles for transport. A consequence of this trend is the existence of traffic jams in the main communication routes of cities [2]. As a result, transit times from one place to another increase [3], the cost of gasoline increases [4], and journeys generally require prior planning and the need to start the journey with additional time to account for possible incidents [5] (accidents, construction, meteorological events, etc.). A particular type of this problem occurs in the airports of large cities [6].

Airports are a critical area for the life of a city, as airports accumulate large numbers of people who arrive in or leave the city on a daily basis. Normally, airports are located on the outskirts of the city, so their access requires transport [7]. In many cases, airports have urban bus lines, underground trains, taxis, and other similar methods of transportation that allow transit from the airport to the city and vice versa. The duration of a trip to access the airport or the city is important [8]. For example, passengers who are going to take a flight need to arrive on time to check in, enter the flight terminal, and reach the departure gate. For passengers travelling to the city from the airport, the duration of the journey is also important. For example, many passengers have planned events, such as meetings, or must take another mode of transport, such as a train, bus, etc. However, the routes between the city and the airport are also affected by the same problems mentioned above [9], and any incident that

occurs on the road will have negative consequences [10]. This work will focus on the journeys that occur from the airport to the city.

When a passenger arrives at the airport and wants to move to the city center, in general, there are two choices: public transport, such as buses, subways, or trains, or private services, such as taxis, private buses, or car rentals. The advantages of public transport [11] is that it is cheap and normally has regular schedules for departures and stops during the journey. However, the main disadvantage [12] is that it is generally slow due to the number of stops. In addition, the objective of public transport is often to bring the passengers to a central location in the city from which the passenger will have to take another means of transportation to reach their final destination. The main advantages of private services are their faster speed compared to public services and that they generally allow passengers to reach their final destination directly without intermediate stops. However, the main disadvantage of such services is that they are more expensive than public services.

A shuttle is a particular type of private bus [13] that has a limited capacity in its number of passengers. Its usual operation is a mix between a bus and taxi, since it usually has a fixed price and takes a passenger to their final destination. In this sense, the planning of the route is manual. When the driver of the shuttle knows all the destinations of the passengers, then he or she can decide upon the necessary routes and stops [14]. To do this, the driver uses his or her knowledge of the city's communication and the average times that each road may require. This information can also be complemented by data obtained from Global Positioning System (GPS) applications [15] or applications that provide information [16] about the state of a city's communication routes (traffic jams, accidents, weather events, and others). The main advantages of shuttle services compared to other modes of transport [17] is that their cost is cheaper than a taxi (the cost is shared by all passengers) and they are faster than a private bus (in addition, a private bus does not normally take passengers to their final destination). However, one disadvantage [18] is the number of different destinations that a shuttle has to visit (in the worst case, one destination per passenger). This situation can produce some negative effects related to [19] (1) the quality of service [20] (such as a delay in reaching the destinations if they are very far from each other or difficulty in estimating when each of the destinations will be reached), (2) the passengers (e.g., passengers not reaching other intermediate means of transport in the city, such as a train, or not arriving on time to a certain event), and (3) economic losses due to not adequately estimating the cost of the journey [21].

A critical aspect for the efficient service of a shuttle is planning and optimizing the route (the selection of the paths to use and the order in which the stops are visited). The planning of an optimized route is called the Travelling Salesman Problem (TSP) [22]. This problem asks the following question: given a list of destinations and the distances between each pair of destinations, what is the shortest possible route that visits each destination exactly once and returns to the origin? There are different approaches to solving this problem: (1) exact algorithms [23] that look for all permutations (ordered combinations) and determine which of these is the smallest (for example, the Held–Karp algorithm [24]) and (2) heuristic algorithms [25] that use some property or information related to the problem to accelerate the solution (for example, the Christofides algorithm [26]).

The original TSP only considers the distances between destinations. However, in the case of a shuttle, the most critical aspect is the travel duration. Thus, the distances do not change, but the durations can change. In this case, the solution of standard TSP maybe not provide an optimized route [27]. In addition, in general, the solution will be dynamic [28] since it can change every day and every moment of the day depending on incidents (accidents, weather, etc.) that occur on the roads [29]. For this reason, there are specific studies that examine TSP in the context of transport in cities, such as [30–34]. These variants of the algorithm are characterized by using probability calculations, exploiting previous knowledge of the domain (normal durations of routes, typical characteristics of traffic on a highway, etc.) or real-time information on the state of the roads (accidents, weather, construction, etc.) to find a model that can predict the best solution.

In recent years, the phenomenon of the internet of things [35] has afforded the possibility of accessing large amounts of information from sensors [36] that are located in different elements that are used daily, such as household appliances, mobile phones, clothing, cars, etc. By processing the information generated by these sensors, it is possible to implement value-added services that can be useful for different types of users and purposes [37]. In particular, there are applications that offer solutions to problems similar to TSP [38] (generally, finding the best route between two places). These applications combine information [39] from mobile geolocation sensors, meteorological information, real-time information on the state of traffic, information from sensors located in the city, and cartographic information on the city. With this information, different routes are calculated between the point of origin and the destination [40], as well as the approximate duration times for each route. Likewise, they offer recommendations on the best route. Some examples of this type of application include Google Maps [41], MapBoox [42], OpenStreetMaps [43], Maps.me [44], Navmii [45], Waze [46], HereWego [47], and MAP3D pro [48]. The main differences between them are in their types of licenses, their sources of information used to perform calculations, their abilities to use the information generated, and additional information on the journey (shops, monuments, and restaurants along the path). Table 1 shows a comparison of these applications.

Table 1. Comparison of products.

Product	License	Use	Updates	Advertising	Online	Accuracy
Google Maps	Proprietary	Free	Fast	Yes	Yes	Very precise
MapBox	Open	Free	Fast	No	Yes	Very precise
OpenStreetMaps	Open	Free	Fast	No	Yes	Very precise
Maps.me	Open	Free	Fast	No	No	Not precise
Navmii	Open	Free	Slow	No	Yes	Precise
Waze	Proprietary	Free	Fast	Yes	Yes	Precise
HereWeGo	Proprietary	Free	Slow	No	Yes	Precise
MAP3D pro	Proprietary	Not free	Fast	Yes	Yes	Precise

These applications have some limitations. First, in general, they are not designed to plan a multi-destination journey (Note that one of the aforementioned applications, MapBox, does allow one to solve the traveler's problem and offers a route showing one how to visit the different destinations of a journey). If the route contains three or more destinations, then it is necessary to decompose the global route into a set of routes between each of the destinations. However, the main problem is that the best order in which to visit each destination cannot be known. This order must be determined manually. Second, these applications are not designed to implement a passenger or journey management system (their purpose is only to provide information on how to move from one place to another).

Related applications are those used by alternative urban transport [49] services, such as Uber [50], Cabify [51], and MyTaxi [52]. These types of applications use the information provided by the applications mentioned above and create a personalized service for their particular needs. In general, these applications allow passenger management, service payment management, route calculations, communication with other employees, and other personalized services. The main limitation [53] of such services is the impossibility of adapting them to different services or reusing their code. In addition, they are not designed to manage routes with several destinations, since the type of service that they offer includes routes with a single destination.

Notably, this paper does not present a new algorithm for the Traveling Salesman Problem (TSP) applied to transport in large cities. This work describes the implementation of an added-value service to automate the planning of a shuttle route to transport passengers from the airport to the center city when there is more than one destination in the route. In addition, the application manages other aspects of the shuttle service, such as route management, passenger management, schedule management, and others. This application combines information from the geolocation sensors of the driver's mobile phone, the information provided by passengers, and the information of the route calculated by the

implementation of a TSP solution offered by the Mapbox API. In this sense, the novelty of this work is its added-value service implemented by combining different sources of information.

The structure of the paper is as follows. In Section 2, the methods and materials used to implement the application are established. Section 3 then discusses the results obtained, and finally, in Section 4, some conclusions and lines of future work are presented.

2. Materials and Methods

2.1. Objectives

Two main objectives were set for this work. The first objective is to automate the planning of the trajectory of a shuttle service to the different destinations of its passengers to optimize the duration of the journey (that is, it is necessary to know the order in which each destination will be visited and the path that will be used). The second objective is to manage the shuttle service process through passenger management, route management, schedule management, and time and destination management.

These objectives can be refined into the following more specific objectives:

- Providing a tool that can calculate the optimal route of a shuttle to the destinations of the passengers. The output generated should indicate in what order the destinations will be visited and which roads should be used. Likewise, the solution provided must be dynamic with respect to specific incidents that may occur and consider them when calculating the route and in the real-time modifications that must be done.
- Facilitating the management of passengers who will use the shuttle service. In this sense, each passenger should be able to register in the application, select a destination for their trip, view the different shuttle services offered by the airport according to schedules and service areas, and determine a priori the estimated time that the journey to their destination will last.
- Facilitating the management of schedules and service areas. In this sense, the shuttle driver should be able to register at what times he or she will offer the service and in which service areas. A problem discussed in the introduction was the enormous distance that can exist between various destinations. To solve this problem, we propose that shuttle services should not serve the entire city. Instead, they should serve only certain areas to guarantee that the duration of the trips to each destination will be more or less homogeneous.
- Facilitating the management of shuttle drivers. This application will allow any driver to register so that they can manage their service.

2.2. Materials

To implement this application, 2 elements were used: the information provided by the mobile geolocation sensors and the information obtained from one of the services offered by the Mapbox service api to calculate the traveler's problem for various destinations.

2.2.1. Geolocation of a Mobile Phone

All smart mobile phones contain a set of sensors that can facilitate geolocation, such as [54]

- Accelerometers [55]. An accelerometer can determine the orientation of the mobile device (horizontal, vertical, face down, or face up). An accelerometer consists of a mobile portion that moves depending on the acceleration applied to it and a fixed part that interprets the voltage resulting from this movement to determine the speed at which it moves and its orientation. These components use three axes to measure motion in a three-dimensional space.
- Barometers. A barometer measures the ambient pressure of the air to determine the height [56]. This helps GPS positioning.

- Gyroscopes [57]. A gyroscope measures non-gravitational acceleration and complements the information that the accelerometer offers on the orientation of the mobile device. To do this, the gyroscope adds a fourth dimension of movement that measures the rotation of the mobile device.
- GPS [58]. A GPS reads the signal transmitted by GPS satellites. In general, GPSs use the intersection angles of at least three satellites with respect to the device to triangulate a person's location. GPSs can also add the distance from the mobile to the mobile phone towers to obtain a more precise location [59].

The Android SDK provides a tool to read the data from the sensors available on a mobile device. This tool is called Sensor Manager [60] and can receive data from sensors with a periodically set sampling period. There are four possible sampling periods: delay_normal (200 ms), delay_ui (60 ms), delay_game (20 ms), and delay_fastest (0 ms). Despite this, sensors such as the accelerometer and the gyroscope are active by default to provide information such as changes in screen orientation. The units used by the Sensor Manager are meters per second squared for the accelerometer and radians per second for the gyroscope. In addition, mobile sensors use the local coordinate system (the local coordinate system is characterized by the fact that the X and Y axes are interchangeable, but normally the X axis points in the longitudinal direction, the Y axis posts in the transverse direction, and the Z axis points towards the vertical direction).

There are different APIs available to locate an Android device [61], including the Android system API, the Google API, and the Fused Location Provider [62]. The latter determines the position through different methods depending on the permissions that the application has and the required precision. It mainly has three priorities: high_accuracy, balanced_power, and no_power. The first requires very precise data, which is why it uses hardware (GPS) to acquire greater precision. The second, depending on data availability, locates the device through cell towers or WIFI. Finally, the third priority is a passive method where the application does not receive position updates but instead takes advantage of requests from other applications. In addition to position updates, the API also provides information on the magnitude of the device's speed and the device's orientation (calculated from the GPS positions).

2.2.2. Mapbox Service

Mapbox is a company that offers online mapping services similar to Google Maps. It takes data from open data sources such as OpenStreetMap and NASA or from private sources such as DigitalGlobe [63]. Among its services, Mapbox offers access to an API of services that can be used with certain limitations. In particular, one of the services offered by the API is the resolution of the traveler problem [64], which returns a route with an optimized duration between a set of geographic coordinates provided. The basic service call has the following structure:/optimized-trips/v1/{profile}/{coordinates}, where profile indicates a type of profile on the means of transport (car, bicycle, walking, etc.), and coordinates provides a list of between 2 to 12 pairs of coordinates (latitude, longitude) that indicate the destinations through which the journey should be taken. Likewise, other optional parameters [64] can be included in the call (see Table 2).

Table 2. Optional parameters.

Parameter	Description
annotations	Return additional metadata along the route. It must include several annotations as a comma-separated list. The possible values are duration (the duration between each pair of coordinates in seconds), distance (the distance between each pair of coordinates in meters), and speed (the speed between each pair of coordinates in meters per second). It requires that the parameter overview = full
approaches	A semicolon-separated list indicating the side of the road from which to approach the waypoints on a requested route. Accepts unrestricted (default, where the route can arrive at the waypoint from either side of the road) or curb data (the route will arrive at the waypoint on the driving_side of the region). If provided, the number of approaches must be the same as the number of waypoints. However, it can skip a coordinate and show its position in the list with the ";" separator. It must be used in combination with steps = true.
bearings	It influences the direction in which a route starts from a waypoint and is used to filter the road segment on which a waypoint will be placed by direction. This is useful for ensuring that rerouted vehicles continue traveling in their current direction. A request that does this will provide the bearing and radius values for the first waypoint and leave the remaining values empty. It must be used in conjunction with the radiuses parameter and takes 2 values per waypoint: an angle clockwise from true north between 0 and 360 and the range of degrees by which the angle can deviate (the recommended value is 45° or 90°), formatted as {angle, degrees}. If provided, the list of bearings must be the same length as the list of waypoints. However, one can skip a coordinate and show its position in the list with the ";" separator.
destination	It specifies the destination coordinate of the returned route and accepts any (default) or the last parameters.
distributions	It specifies the pick-up and drop-off locations for a trip by providing a delimited list of numbered pairs that correspond to the coordinate list. The first number of a pair indicates the index for the coordinate of the pick-up location in the coordinate list, and the second number indicates the index for the coordinate of the drop-off location in the coordinate list. Each pair must contain exactly 2 numbers, which cannot be the same. The returned solution will visit the pick-up locations before visiting the drop-off locations. The first location can only be a pick-up location, not a drop-off location.
geometries	The format of the returned geometry. The allowed values are geojson (as LineString), polyline (default, a polyline with precision 5), and polyline6 (a polyline with precision 6).
language	The language of the returned turn-by-turn text instructions. The default is en (English).
Overview	The type of the returned overview geometry. It can be full (the most detailed geometry available), simplified (default, a simplified version of the full geometry), or false (no overview geometry).
Radiuses	The maximum distance a coordinate can be moved to snap to the road network in meters. There must be as many radiuses as there are coordinates in the request, each separated by ";". Values can be any number greater than 0 or an unlimited string. A NoSegment error is returned if no routable road is found within the radius.
source	The coordinate at which to start the returned route. Accepts any (default) or first parameters.
steps	Determines whether to return the steps and turn-by-turn instructions (true) or not (false, default).
roundtrip	Indicates whether the returned route is a roundtrip, meaning that the route returns to the first location (true, default) or not (false). If roundtrip = false, the source and destination parameters are required, but not all combinations will be possible.

The request returns a JSON document containing information on the calculated path:

- ode. A string indicating the state of the response. This is a separate code from the HTTP status code. On normal valid responses, the value will be "Ok".
- waypoints. An array of waypoint objects. Each waypoint is an input coordinate snapped to the road and path network. The waypoints appear in the array in the order of the input coordinates. The content of each waypoint is as follows: name (a string with the name of the road or path that the input coordinate snapped to), location (an array containing the [longitude, latitude] of the snapped coordinate), trips_index (the index of the trip object that contains this waypoint in the trip array), and waypoint_index (the index of the position of a given waypoint within the trip).
- trips. An array of 0 or 1 trip objects. A trip object describes a route through multiple waypoints. The content of each trip object is as follows: geometry (depending on the geometries parameter, which is either a GeoJSON LineString or a Polyline string. Depending on the overview parameter, this is the complete route geometry (full), simplified geometry to the magnified level at which the route can be displayed in full (simplified), or not included (false)); legs (an array of route leg objects); weight_name (a string indicating which weight was used. The default is routability, which is duration-based, with additional penalties for less desirable maneuvers); weight (a float indicating the weight in the units described by weight_name); duration (a float indicating the estimated travel time in seconds); and distance (a float indicating the distance traveled in meters).

2.3. Architecture and Data Model

According to the previous sources of information, a system with this structure is shown in Figure 1.

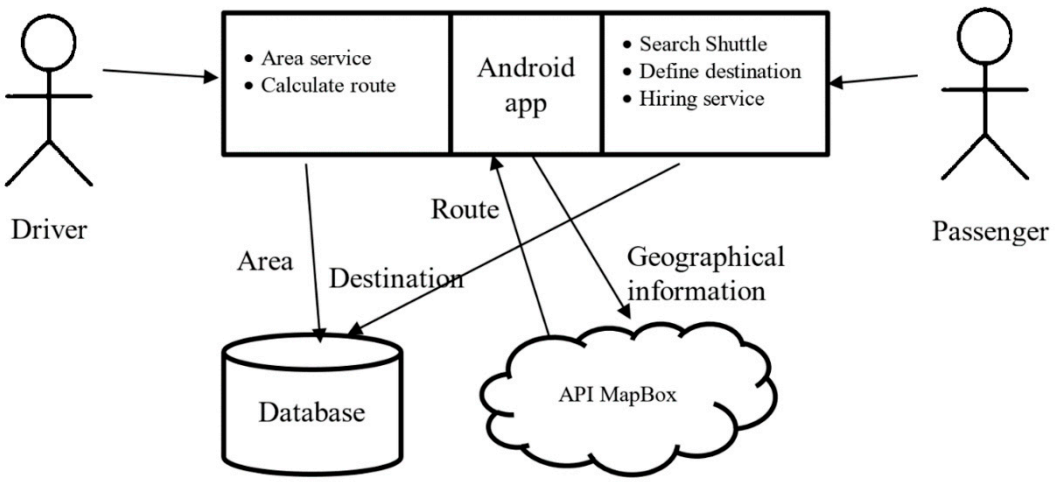

Figure 1. Structure of the solution.

In this system, three actors are defined: the administrator, the driver, and the passenger. The main functions of the driver are the configuration of the shuttle service area and the service hours. A shuttle will not offer service to the entire city, only to a particular area to avoid two destinations that are overly far from each other. Likewise, the service of each shuttle will be carried out at times that the driver will also configure. The passenger will be able to search for the shuttle services registered at an airport and set their desired destinations using a map where they can select this service. In addition, one will be able to carry out the process of contracting the service and managing all the information on the journey, including the route, approximate duration of the trip, and others. Finally, the administrator's role is to perform application maintenance operations. The operation of the system is as follows. The information provided by the driver and each passenger is stored in an online database. When the journey is about to start, and the final list of destinations is already available, the driver executes a script that takes the destination data and geographic information from the airport and invokes the MapBox api optimization service. The service returns a json document with detailed information on the optimal path for the driver to follow. The system reads the information returned and plans

the route, showing the driver the way to go. The driver follows the instructions provided by the application to travel to each destination. For this, the driver's geolocation is also used to guide him or her through the generated route. If there is a specific incident during the journey, the driver can invoke the route calculation service again and update the route to find a better path than the original one. Moreover, all the information on the route and the approximate times of arrival at each destination are accessible by each of the passengers.

To implement the system, a client-server architecture is used, as shown in Figure 2.

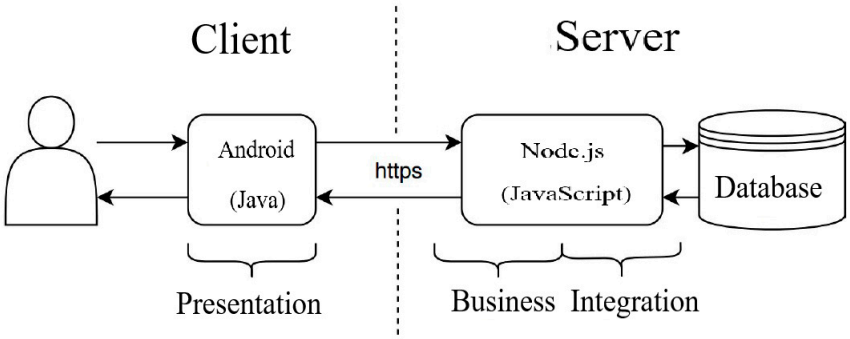

Figure 2. Architecture of the solution.

The role of the client is played by an Android application, which interacts with the server through https calls. A set of services has been implemented on the server using node.js, which are invoked from the Android application. One of the server's tasks is to interact with the database to retrieve and store information. The database was implemented using the Firebase online database. This database provides information on persistent services for web applications and mobile applications, such as

- Realtime database/Cloud firestore: stores and synchronizes data in one of these two databases in the JSON format. In addition, it is possible to add triggers for different types of requests.
- Authentication: a service that facilitates login and its management.
- Cloud Storage: a service that allows the storage and retrieval of files from an application.
- Cloud functions: a service in charge of managing the server code executed when an https request is received.

Specifically, a data model is defined with the structure shown in Figure 3.

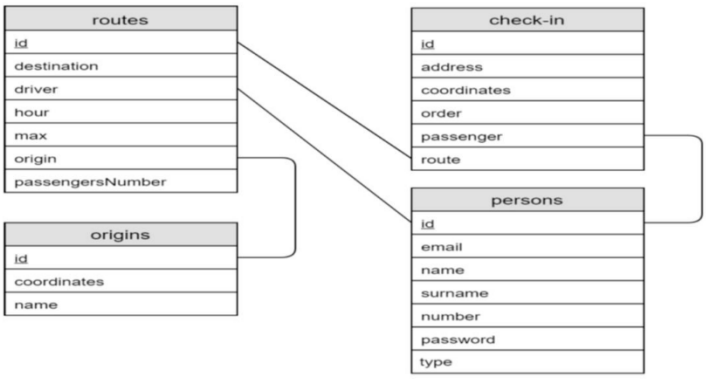

Figure 3. Data model.

The following describes the fields of the types of documents that are stored in the database:

1. persons: This stores the relevant data of the people who use this application. The data stored in the documents are the same regardless of the type of user, as the attributes do not vary. The fields of the documents include:

- email: The email must be unique for each person, although it is not used as a differentiating element among users.
- name: The name of the user.
- surname: The surname of the user.
- number: The telephone number.
- password: The password required to enter the application.
- type: The user type of the application: "passenger", "driver", and "admin". This attribute is used to determine if a certain user has permission to perform a certain request or access a specific view.

2. origins: This stores the origins entered by the administrator. The fields of the documents are:

   - coordinates: The coordinates of the origin.
   - name: The name of the street, square, terminal, etc. from where the bus will depart.

3. routes: This contains information on the current routes. The fields of the documents are:

   - destination: This is determined by postal code. The driver chooses the postal code of the area where it will perform its stops. In this way, passengers whose destinations are within that code may be included in the trip.
   - driver: The identification of the driver who will make the journey.
   - hour: The departure time. This attribute is used so that once the passenger has selected an origin and a destination, a list with all the journeys and departure times can be displayed, and the passenger can choose the most convenient one.
   - max: The maximum number of passengers that the shuttle bus can take. When the current number of passengers equals this number, no further reservations are allowed.
   - origin: The identification of the origin of the route.
   - passenger numbers: The number of passengers who have registered for the trip.

4. check-in: This saves the elements that the passengers provide when adding themselves to a journey. Using this table, it is possible to recover the trips to which a passenger is registered. The fields of this document are:

   - address: Full address where the passenger wants to get off.
   - coordinates: The coordinates of the place where the passenger wants to get off.
   - order: The position on the bus list.
   - passenger: The passenger's identification.
   - route: Identification of the route to which the passenger has registered.

Note that to organize the data and avoid inconsistencies, the IDs automatically generated by Firebase are used as identifiers. In most cases, they are used to reference data between different collections.

3. Results

The implemented application presents three different interfaces depending on the type of user. The main functionalities of each interface are described below.

3.1. Administrator Interface

The administrator is responsible for establishing the point of origin of a route. The interface consists of two views. The first view (see Figure 4a) contains a map and a search bar in which the point of origin can be searched. For this process, one's name is entered into the bar, which will locate the place on the map. Manually, it is possible to manipulate the search on the map to refine the place

being searched. Then, in the second view (see Figure 4b), the administrator will enter the name of the point of origin. When both steps have been completed, the process is finished by clicking on the add button ("AÑADIR ORIGEN"). If the process has been carried out correctly, a new window appears (see Figure 4c), in which the administrator can use the delete button ("ELIMINAR") to delete the created origin or the edit button ("EDITAR") to modify the name of the origin.

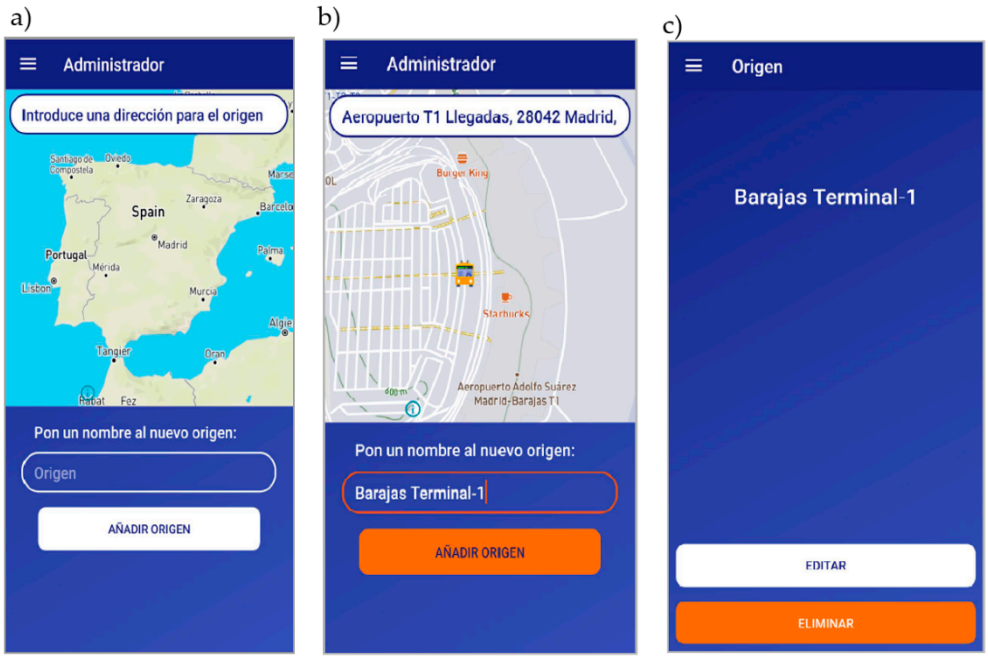

Figure 4. (**a**) Administrator interface, (**b**) selection origin, and (**c**) origin interface.

In the upper left part of the interface, there is a menu with several additional functions (see Figure 5a), including:

- The start link ("Inicio"), which redirects to the main interface.
- The origins link ("Orígenes"), which allows access to a list of all the points of origin registered in the application (see Figure 5b). When one clicks on any point of origin, then the previous functions can be accessed to delete or edit them.
- The configuration link ("Ajustes"), which allows one to configure some aspects of the application.
- The link to close the session ("Cerrar Sesión"), which allows one to close the user session.

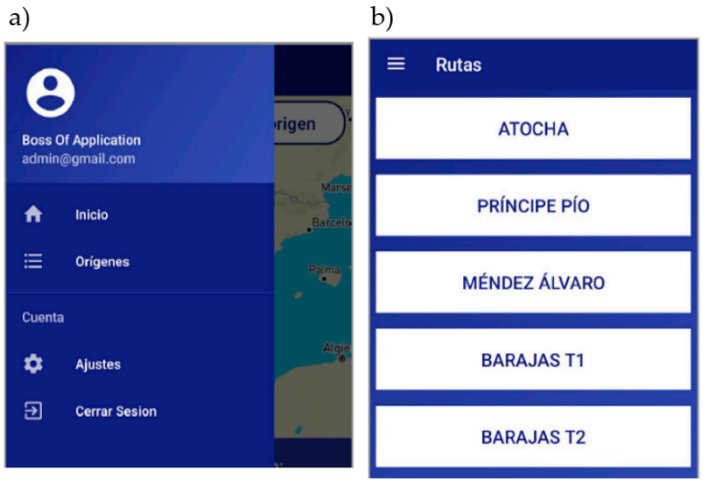

Figure 5. (**a**) Administrator's menu and (**b**) list of origins registered.

3.2. Driver Interface

The driver is responsible for creating new routes. The main driver interface (See Figure 6a) consists of a form in which the data corresponding to the route must be entered, including the point of origin, the postal code where the service is to be performed (the service is limited to a postal code to avoid long distances between two different destinations), the maximum number of passengers, and the departure time from the point of origin. Next, the create route button ("CREAR TRAYECTO") must be clicked. If the process has been carried out correctly, a new interface is displayed (see Figure 6b), which shows the route information and several options, including a button to delete ("ELIMINAR") the route (if there are no passenger requests for that route) and the option to start ("COMENZAR") the route (if the number of passenger requests for that route is greater than or equal to 1). If the option to start the trip is clicked, then the application retrieves the destinations of the passengers and calculates the optimal route using the Mapbox API. The route is shown below on the map, and the option to start the journey is provided. When the option to start the trip is clicked, then the map zooms in (See Figure 6c) and points to the position on the map where the driver is located; next, the application will begin to guide the driver's movements. If an incident occurs, the driver can return to the previous interface and recalculate the optimal route (which generates a new call to the Mapbox api). In this case, the passenger's information on their trip to the given destination is also updated.

Figure 6. (**a**) Main driver interface, (**b**) interface of route created, and (**c**) route.

Likewise, in the upper left part of the interface, there is a menu with several additional functions (see Figure 7a):

- The start link ("Inicio"), which redirects to the main interface.
- The route link ("Rutas"), which allows access to a list with all routes registered by the driver (see Figure 7b). When the driver clicks on any route, then the previous functions to delete or start are accessed.
- The configuration link ("Ajustes"), which allows one to configure some aspects of the application.
- The link to close the session ("Cerrar Sesión"), which allows one to close the user session.

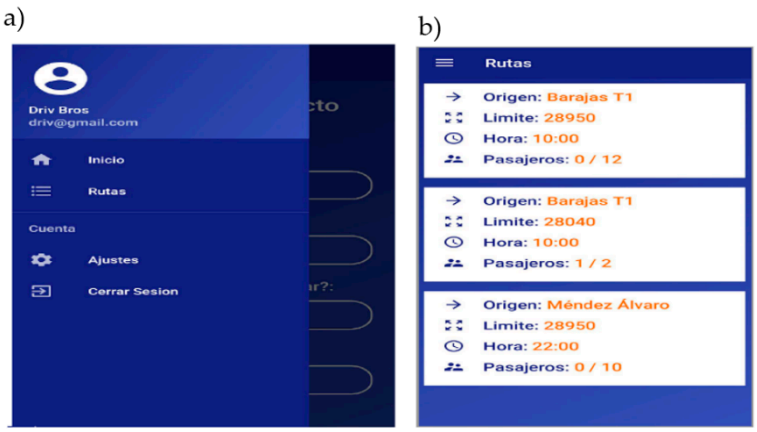

Figure 7. (**a**) Menu and (**b**) list of the available routes.

3.3. Passenger Interface

The passenger is a user who hires a shuttle service published in the application. The main interface (See Figure 8a) consists of a map that shows the user's geolocation, through which it is possible to navigate. At the top of this map, there are two search bars in which the passenger must enter their point of origin (from a list of possible points of origin) and their destination. When the origin is selected, an icon of a bus is added to the map at the coordinates of the indicated address, and when the destination is entered, an icon in the form of a flag is added. In both cases, the map is enlarged so that the passenger can confirm the address.

Figure 8. (**a**) Main passenger interface, (**b**) list of bus routes, and (**c**) details of the route.

Each time the user types in the search bars, suggestions appear for automatic completion. To do this, for each complete word entered by the user, a request is launched through the Mapbox API that returns a list of addresses that match the word inserted by keyboard. This list is used for autocompletion. Likewise, in the same application, the coordinates are used to mark the map and calculate the routes. Next, the passenger clicks on the route search button ("BUSCAR TRAYECTO"), which shows a list with the different shuttle routes and schedules (See Figure 8b) that are available according to the data entered. When the passenger selects a route from the list, its details are displayed (See Figure 8c), and the option of booking a trip is offered by clicking on the add button ("AÑADIR").

Likewise, in the upper left part of the interface, there is a menu with several additional functions (see Figure 9a):

- The start link ("Inicio"), which redirects to the main interface.
- The route link ("Rutas"), which allows access to a list with all routes contracted by the passenger. (see Figure 9b). When the passenger is on a route from the list, the information for that route is displayed (see Figure 9c). In addition, a link is shown to cancel the route.
- The configuration link ("Ajustes"), which allows one to configure some aspects of the application.
- The link to close session ("Cerrar Sesión"), which allows one to close the user session.

Figure 9. (**a**) Menu, (**b**) list of the available routes, and (**c**) details of the route.

3.4. Other Interfaces

Regardless of the type of user, there is a set of common interfaces (see Figure 10):

- In the main interface, the user can be authenticated by entering a username and password.
- In the registration interface, the user can create an account in the application. To do this, the user must enter an email, contact telephone number, name and surname, username and password, and type of user. It is only possible to register as a driver or passenger. The admin user type has its own username and password.

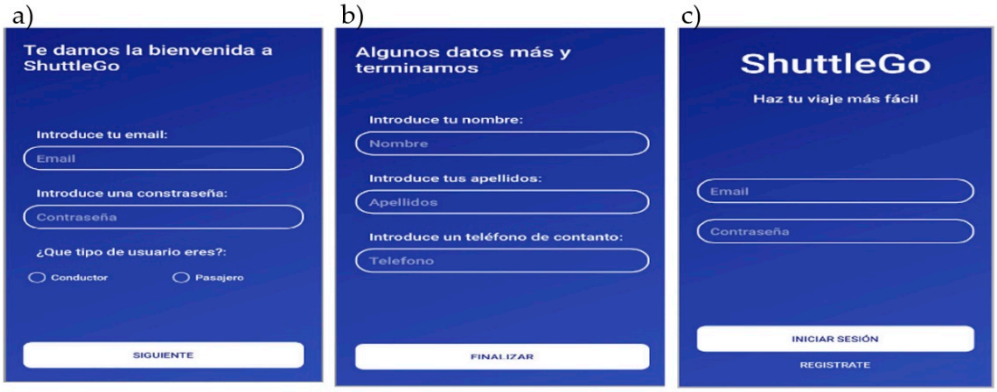

Figure 10. (**a**) Register, (**b**) register, and (**c**) login.

3.5. Evaluation

The usability of the application was evaluated among a group of 20 people, among which 70% were between 19 and 35 years old, and the remaining 30% were older than 36 years. The evaluation consisted of executing a set of tasks in the application acting as a driver (registering as a new driver, logging in as a driver, registering a new route, and canceling a trip) and as a passenger (registering as a new passenger, logging in as a passenger, searching for a trip, and booking or cancelling a reservation).

After completing the tasks, the evaluators answer a set of questions about the usability of the app using Google Forms. The scores for all the questions are on a scale from 0 to 5, with 5 indicating the highest satisfaction and 0 being the lowest.

Figure 11 shows the results for some of the questions asked:

- Utility of the application: 13 people provided a satisfaction level of 4 points, 2 people provided 3 points, and the remaining 5 provided it with 5 points.
- Ease of use of the application: 7 people answered the question with 5 points, 8 people provided 4 points, and the remaining 5 people provided 3 points.
- Application design: 55% of the respondents rated it as "very good", 25% rated it as "good", and the remaining 20% viewed it as "excellent".
- Overall evaluation: 55% of those surveyed rated it as "very good", 30% considered it "excellent", and the remaining 15% rated it as "good".

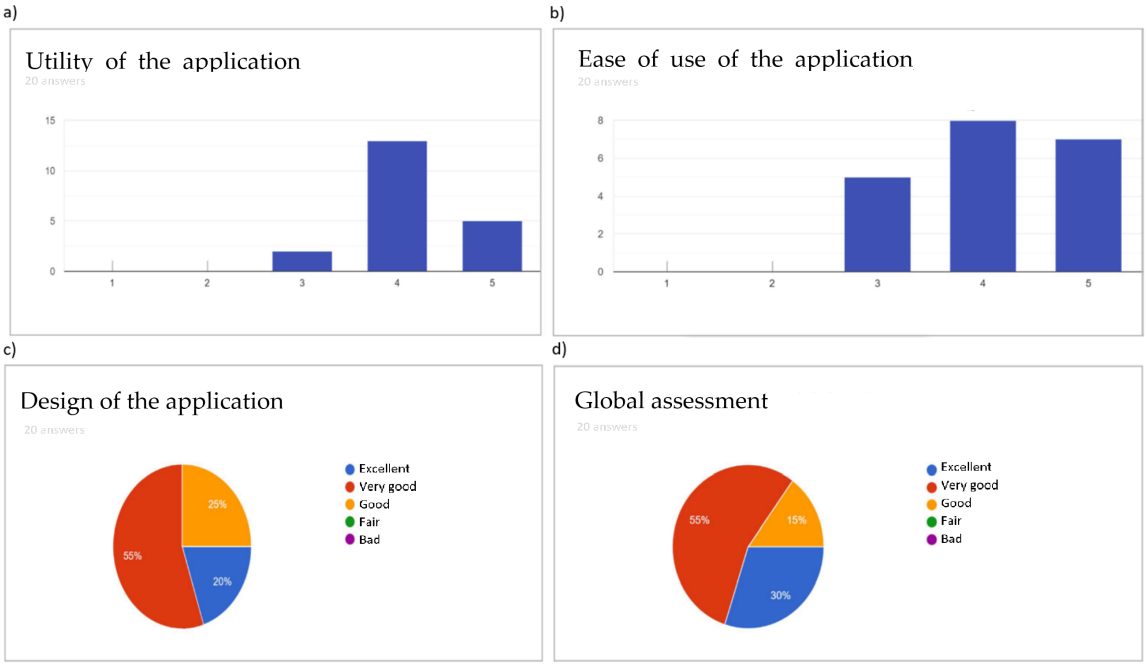

Figure 11. (**a**) Utility of the application, (**b**) Ease of use of the application, (**c**) Design of the application, and (**d**) Global assessment.

4. Conclusions and Future Work

The present study described an application that helps to manage some aspects of the services offered by shuttles at airports. To do this, a value-added service was implemented that combines geolocation information from the driver's mobile phone, passenger information, and uses a service of the MapBox API that calculates the optimal route to solve the TSP problem. In addition, this application allows drivers to manage passengers, schedules, and routes. Although this application is designed for shuttles that operate in airports, it can also be used in other contexts. In this sense, it can be easily adapted for other types of private passenger transport, such as Uber, Cabify, etc., as well as for places that are not airports, such as train stations or bus stations. Likewise, it would be feasible to replace the Mapbox service with other implementations that solve the TSP problem, since this application only integrates the solution and treats the Mapbox service as a black box.

The application partially covers some of the needs that are required to manage a shuttle service. However, there are some lines of future work needed to improve the application, including the following:

- Providing more information to the user about the trip they are taking.
- Improving the system to delimit the service area.
- Implementing a system to encrypt the personal information of users.
- Managing the payment of trips.
- Implementing forums for evaluating the service received, as well as the possibility of recommending both the service and the driver.
- Implementing a messaging system to allow communication between the passengers and the driver.

Funding: This work was supported by the Research Project CetrO+Spec (TIN2017-88092-R).

Conflicts of Interest: The author declares no conflict of interest.

References

1. Akabane, A.T.; Gomes, R.L.; Pazzi, R.W.; Madeira, E.R.M.; Villas, L.A. APOLO: A Mobility Pattern Analysis Approach to Improve Urban Mobility. In Proceedings of the 2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, USA, 2017; pp. 1–6.
2. Dubey, P.P.; Borkar, P. Review on techniques for traffic jam detection and congestion avoidance. In Proceedings of the 2015 2nd International Conference on Electronics and Communication Systems (ICECS), Coimbatore, India, 26–27 February 2015; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, USA, 2015; pp. 434–440.
3. Vencataya, L.; Pudaruth, S.; Dirpal, G.; Narain, V. Assessing the Causes & Impacts of Traffic Congestion on the Society, Economy and Individual: A Case of Mauritius as an Emerging Economy. *Stud. Bus. Econ.* **2018**, *13*, 230–242. [CrossRef]
4. Marshall, W.E.; Dumbaugh, E. Revisiting the relationship between traffic congestion and the economy: A longitudinal examination of U.S. metropolitan areas. *Transportation* **2018**, *47*, 275–314. [CrossRef]
5. Islam, A. A Literature Review on Freeway Traffic Incidents and Their Impact on Traffic Operations. *J. Transp. Technol.* **2019**, *9*, 504–516. [CrossRef]
6. Benevolo, C.; Dameri, R.P.; D'Auria, B. Smart Mobility in Smart City. In *Lecture Notes in Information Systems and Organisation*; Springer Science and Business Media LLC: Berlin/Heidelberg, Germany, 2015; Volume 11, pp. 13–28.
7. Nuzzolo, A.; Comi, A.; Polimeni, A. Exploring on-demand service use in large urban areas: The case of Rome. *Arch. Transp.* **2019**, *50*, 77–90. [CrossRef]
8. Jayasooriya, S.; Bandara, Y. Measuring the Economic costs of traffic congestion. In Proceedings of the 2017 Moratuwa Engineering Research Conference (MERCon), Moratuwa, Sri Lanka, 29–31 May 2017; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, USA, 2017; pp. 141–146.
9. Jacyna, M. The urban logistic service model in the aspect of the domestic logistic system. In *Urban Transport XVII: Urban Transport and the Environment in the 21st Century*; IEEE: Piscataway, NJ, USA, 2011; pp. 116–117.
10. Dadashova, B.; Li, X.; Turner, S.; Koeneman, P. Multivariate time series analysis of traffic congestion measures in urban areas as they relate to socioeconomic indicators. *Socio-Econ. Plan. Sci.* **2020**, 100877. [CrossRef]
11. Da Costa, D.C.T.; De Neufville, R. Designing Efficient Taxi Pickup Operations at Airports. *Transp. Res. Rec. J. Transp. Res. Board* **2012**, *2300*, 91–99. [CrossRef]
12. Meneguette, R.I.; Filho, G.P.R.; Bittencourt, L.F.; Ueyama, J.; Krishnamachari, B.; Villas, L.A. Enhancing intelligence in inter-vehicle communications to detect and reduce congestion in urban centers. In Proceedings of the 2015 IEEE Symposium on Computers and Communication (ISCC), Natal, Brazil, 25–28 June 2018; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, USA, 2015; pp. 1–6.
13. Muller, P.J.; Sproule, W.J. Airport Parking Shuttle Comparison: Bus vs. Personal Rapid Transit. In *Automated People Movers and Automated Transit Systems 2016*; American Society of Civil Engineers (ASCE): Toronto, ON, Canada, 2016; pp. 63–73.
14. Guo, X.; Song, R.; He, S.; Bi, M.; Jin, G. Integrated Optimization of Stop Location and Route Design for Community Shuttle Service. *Symmetry* **2018**, *10*, 678. [CrossRef]

15. Liqun, L.; Chaozhong, W.; Hui, Z.; Hasan, A.H.N.; Wenhui, C.; Charles, A. Research on taxi drivers' passenger hotspot selecting patterns based on GPS data: A case study in Wuhan. In Proceedings of the 2017 4th International Conference on Transportation Information and Safety (ICTIS), Banff, AB, Canada, 8–10 August 2017; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, USA, 2017; pp. 432–441.
16. Yao, B.; Cao, Q.; Jin, L.; Zhang, M.; Zhao, Y. Circle Line Optimization of Shuttle Bus in Central Business District without Transit Hub. *Promet-Traffic Transp.* **2017**, *29*, 45–55. [CrossRef]
17. Ralphs, E.; Shahab, S.; Ahmadpoor, N.; Emma, R.; Sina, S.; Negar, A. Access to Small Airports and the Impact on Regional Growth in the UK. *Curr. Urban Stud.* **2020**, *8*, 24–56. [CrossRef]
18. Cao, Y.; Wang, J. The Key Contributing Factors of Customized Shuttle Bus in Rush Hour: A Case Study in Harbin City. *Procedia Eng.* **2016**, *137*, 478–486. [CrossRef]
19. Xiao, Q.; He, R.; Ma, C. The analysis of urban taxi carpooling impact from taxi GPS data. *Arch. Transp.* **2018**, *47*, 109–120. [CrossRef]
20. Loi, L.T.I.; So, A.S.I.; Lo, I.S.; Fong, L.H.N. Does the quality of tourist shuttles influence revisit intention through destination image and satisfaction? The case of Macao. *J. Hosp. Tour. Manag.* **2017**, *32*, 115–123. [CrossRef]
21. Liu, Y.; Jia, G.; Tao, X.; Xu, X.; Dou, W. A Stop Planning Method over Big Traffic Data for Airport Shuttle Bus. In Proceedings of the 2014 IEEE Fourth International Conference on Big Data and Cloud Computing, Sydney, Australia, 3–5 December 2014; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, USA, 2014; pp. 63–70.
22. Raman, V.; Gill, N.S. Review of different heuristic algorithms for solving Travelling Salesman Problem. *Int. J. Adv. Res. Comput. Sci.* **2017**, *8*, 423–425. [CrossRef]
23. Laporte, G. The traveling salesman problem: An overview of exact and approximate algorithms. *Eur. J. Oper. Res.* **1992**, *59*, 231–247. [CrossRef]
24. Valenzuela, C.L.; Jones, A.J. Estimating the Held-Karp lower bound for the geometric TSP. *Eur. J. Oper. Res.* **1997**, *102*, 157–175. [CrossRef]
25. Wu, Y.; Song, W.; Cao, Z.; Zhang, J.; Lim, A. Learning Improvement Heuristics for Solving Routing Problems. Available online: https://arxiv.org/pdf/1912.05784.pdf (accessed on 11 September 2020).
26. Nilsson, C. *Heuristics for the Traveling Salesman Problem*; Linkoping University: Linkoping, Sweden, 2003.
27. Yazici, A.; Kamga, C.; Singhal, A. Modeling taxi drivers' decisions for improving airport ground access: John F. Kennedy airport case. *Transp. Res. Part A Policy Pract.* **2016**, *91*, 48–60. [CrossRef]
28. Yazici, M.A.; Kamga, C.; Singhal, A.; Singhal, A. A big data driven model for taxi drivers' airport pick-up decisions in New York City. In Proceedings of the 2013 IEEE International Conference on Big Data, Santa Clara, CA, USA, 6–9 October 2013; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, USA, 2013; pp. 37–44.
29. Jacyna, M.; Wasiak, M.; Kłodawski, M.; Gołębiowski, P. Modelling of Bicycle Traffic in the Cities Using VISUM. *Procedia Eng.* **2017**, *187*, 435–441. [CrossRef]
30. Cao, Z.; Guo, H.; Zhang, J. A Multiagent-Based Approach for Vehicle Routing by Considering Both Arriving on Time and Total Travel Time. *ACM Trans. Intell. Syst. Technol.* **2018**, *9*, 1–21. [CrossRef]
31. Cao, Z.; Guo, H.; Song, W.; Gao, K.; Chen, Z.; Zhang, L.; Zhang, X. Using Reinforcement Learning to Minimize the Probability of Delay Occurrence in Transportation. *IEEE Trans. Veh. Technol.* **2020**, *69*, 2424–2436. [CrossRef]
32. Cao, Z.; Wu, Y.; Rao, A.; Klanner, F.; Erschen, S.; Chen, W.; Zhang, L.; Guo, H. An Accurate Solution to the Cardinality-Based Punctuality Problem. *IEEE Intell. Transp. Syst. Mag.* **2018**, *3*, 145–160. [CrossRef]
33. Cao, Z.; Guo, H.; Zhang, J.; Niyato, D.; Fastenrath, U. Finding the Shortest Path in Stochastic Vehicle Routing: A Cardinality Minimization Approach. *IEEE Trans. Intell. Transp. Syst.* **2015**, *17*, 1688–1702. [CrossRef]
34. Cao, Z.; Guo, H.; Zhang, J.; Niyato, D.; Fastenrath, U. Improving the Efficiency of Stochastic Vehicle Routing: A Partial Lagrange Multiplier Method. *IEEE Trans. Veh. Technol.* **2015**, *65*, 3993–4005. [CrossRef]
35. Ray, P.P. A survey on Internet of Things architectures. *J. King Saud Univ.-Comput. Inf. Sci.* **2018**, *30*, 291–319. [CrossRef]
36. Lopez, J.; Rios, R.; Bao, F.; Wang, G. Evolving privacy: From sensors to the Internet of Things. *Futur. Gener. Comput. Syst.* **2017**, *75*, 46–57. [CrossRef]
37. Kang, J.J.W.; Larkin, H. Inference of Personal Sensors in Internet of Things. *Int. J. Inf. Commun. Technol. Appl.* **2016**, *2*, 1–23. [CrossRef]

38. Kamga, C.; Conway, A.; Singhal, A.; Yazici, A. Using Advanced Technologies to Manage Airport Taxicab Operations. *J. Urban Technol.* **2012**, *19*, 23–43. [CrossRef]
39. Li, Y.; Lu, J.; Zhang, L.; Zhao, Y. Taxi Booking Mobile App Order Demand Prediction Based on Short-Term Traffic Forecasting. *Transp. Res. Rec. J. Transp. Res. Board* **2017**, *2634*, 57–68. [CrossRef]
40. Chuah, S.P.; Wu, H.; Lu, Y.; Yu, L.; Bressan, S. Bus Routes Design and Optimization via Taxi Data Analytics. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, Indianapolis, IN, USA, 24–28 October 2016; pp. 2417–2420.
41. Gibson, R.; Erle, S. *Google Maps Hacks*; O'Reilly Media, Inc.: New York, NY, USA, 2006; pp. 25–45.
42. Kastanakis, B. *Mapbox Cookbook*; Packt Publishing Ltd.: Birmingham, UK, 2016; pp. 61–72.
43. Bennett, J. *OpenStreetMap*; Packt Publishing Ltd.: Birmingham, UK, 2016; pp. 45–68.
44. Mapsme. Available online: https://maps.me (accessed on 17 August 2020).
45. Navmii. Available online: https://www.navmii.com/ (accessed on 17 August 2020).
46. Waze. Available online: https://www.waze.com (accessed on 17 August 2020).
47. HereWeGo. Available online: https://here.com/app (accessed on 17 August 2020).
48. Map3D pro. Available online: https://www.map3d.com/ (accessed on 17 August 2020).
49. Navarro, N.A.G. Vehicle rental with driver (VTC) and its legal implications. Uber, Cabify and the colaborative economy. *Rev. Estud. Adm. Local Autonómica* **2018**, *9*, 128–140.
50. Uber. Available online: https://www.uber.com (accessed on 17 August 2020).
51. Cabify. Available online: https://cabify.com (accessed on 17 August 2020).
52. MyTaxi. Available online: https://free-now.com/ (accessed on 17 August 2020).
53. Rayle, L.; Shaheen, S.A.; Chan, N.; Dai, D.; Cervero, R. *App-Based, on-Demand Ride Services: Comparing Taxi and Ridesourcing Trips and User Characteristics in San Francisco (No. UCTC-FR-2014-08)*; University of California Transportation Center: Berkeley, CA, USA, 2014.
54. Yang, Z.; Wu, C.; Zhou, Z.; Zhang, X.; Wang, X.; Liu, Y. Mobility Increases Localizability. *ACM Comput. Surv.* **2015**, *47*, 1–34. [CrossRef]
55. Jain, A.; Kanhangad, V. Human Activity Classification in Smartphones Using Accelerometer and Gyroscope Sensors. *IEEE Sens. J.* **2018**, *18*, 1169–1177. [CrossRef]
56. Al-Turjman, F. Impact of user's habits on smartphones' sensors: An overview. In *2016 HONET-ICT*; IEEE: Piscataway, NJ, USA, 2016; pp. 70–74.
57. Li, F.; Zhao, C.; Ding, G.; Gong, J.; Liu, C. A Reliable and Accurate Indoor Localization Method Using Phone Inertial Sensors. In Proceedings of the 2012 ACM Conference on Computer and Communications Security—CCS '12, New York, NY, USA, 5–8 September 2012; p. 421.
58. Deng, Z.-A.; Hu, Y.; Yu, J.; Na, Z. Extended Kalman Filter for Real Time Indoor Localization by Fusing WiFi and Smartphone Inertial Sensors. *Micromachines* **2015**, *6*, 523–543. [CrossRef]
59. Liu, Y.; Dashti, M.; Rahman, M.A.A.; Zhang, J. Indoor localization using smartphone inertial sensors. In Proceedings of the 2014 11th Workshop on Positioning, Navigation and Communication (WPNC), Dresden, Germany, 11–12 March 2010; IEEE: Piscataway, NJ, USA, 2014; pp. 1–6.
60. Milette, G.; Stroud, A. *Professional Android Sensor Programming*; John Wiley & Sons: Hoboken, NJ, USA, 2012; pp. 87–98.
61. Zhi-An, Y.; Chun-Miao, M. The development and application of sensor based on android. In Proceedings of the 2012 8th International Conference on Information Science and Digital Content Technology (ICIDT2012), Jeju, Korea, 26–28 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 231–234.
62. Gurram, B.; Giri, N. Improving localization accuracy of android's Fused Location Provider API using Kalman Filter. In Proceedings of the 2016 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 7–9 January 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–4.
63. van Rees, E. DigitalGlobe and big data. *GeoInformatics* **2016**, *19*, 6.
64. Saputra, K.; Furqan, M.; Abidin, T.F.; Yunadi, D.H. Google maps and mapbox api performance analysis on android-based lecture attendance application. *J. Nat.* **2019**, *19*, 64–68. [CrossRef]