DRIVES: Android App for Automotive Customized Services

Marco De Vincenzi
IIT, Consiglio Nazionale delle Ricerche
Pisa, Italy
marco.devincenzi@iit.cnr.it

2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA) | 979-8-3503-1008-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/AICCSA56895.2022.10017757

***Abstract*—Today, Big Data, generated by connected vehicles, can improve road safety, lead the green transition, and direct the mobility revolution. For this reason, we develop an Android app, called DRIVES (DRIVEr Services), that allows drivers to get rewarded for their eco-driving style. In particular, DRIVES collects vehicle data from the OBD-II port, stores them in infrastructure of the European project E-Corridor, computes a driving style profile, called Driver DNA, and provides users customized services, based on the driving profiles. The offered services can span a wide range of possibilities from an energy bill discount to a personalized carsharing price, but, in any case, as a direct consequence of the rewarding process, our app can encourage safer and more sustainable mobility.**

In this work, we describe the app structure, its operations, and the E-Corridor architecture, where vehicle data are analyzed. Besides, we provide an insight into the possible privacy concerns caused by the usage of personal and vehicle data. The result is an Android app that has been tested in the project infrastructure with a real vehicle, and that can be used as a general schema to create other rewarding apps, also for autonomous vehicles, using driving data.

***Index Terms*—Automotive, Android, Vehicle Services, Eco-driving, Privacy.**

I. Introduction

In the last years, the automotive industry is facing significant transitions with not only the design of low-emission vehicles but also the increase of digitization and automation. The growing connected road infrastructure is offering possibilities unthinkable until a few years ago. For instance, vehicles can communicate directly with the Roadside Units (RSUs), which are transceivers mounted along a road or passageway to transfer data and information. The possibility to collect data from vehicles and roads, can increase efficiency and lower the environmental costs of mobility. In this scenario, we develop an Android app that can exploit the potential of data sharing and, then, contribute to the efficiency of the road ecosystem.

Our Android app, called DRIVES (DRIVEr Services), retrieves data from the vehicles and allows drivers to receive customized services, based on their driving style. As *"driving style"* we mean a personal driver's usual method of driving, defined using several vehicle parameters in a long-term driving [1]. With the app, the drivers can live a full personalized driving experience during their journeys, receiving customized services like ad-hoc recharging or restaurant prices, or suggestions for the most suitable road, according to the users' habits and needs. As a consequence, at the same time, the app reward system can motivate drivers to modify their driving styles to receive benefits.

DRIVES can work on every device equipped with an Android Operating System (OS) like smartphones or the infotainment of a vehicle. The app is driver-friendly with an intuitive interface so, during the driving, it enables users to ask for customized services. DRIVES is designed into the European project *E-Corridor* and it works with its infrastructure. E-Corridor aims to develop a technological framework to unleash the power of information sharing, coupled with edge-based collaborative analytics for cyber protection.

DRIVES uses vehicle data, collected from the OBD-II port to define the driver's driving style, and it uses a metric, called Driver DNA [2], which is based on four main parameters: breaking, turning, speeding, and revolutions per minute (RPM). The app sends data to the E-Corridor server, which analyzes them, asks a service provider (SP), like a charging station or a restaurant, for a specific service, and returns the required information to the driver. With this schema, the driver is encouraged to follow the indications that the app provides to reduce consumption and risks. To the best of our knowledge, DRIVES is one of the first solutions which combines data collected from a vehicle, with the target to increase the efficiency of road mobility to assure higher safety and go greener. Besides, DRIVES was tested on a real vehicle and it could be ready to be applied in a complete Intelligent Transportation System (ITS) infrastructure.

A. Structure of the paper

Section II describes the related work, in particular the apps which retrieve data from the OBD-II port and the apps which provide drivers personalized services. Section III summarizes the main components on which our app is based so it describes the *Driver DNA*, the E-Corridor infrastructure, and the Pires' library, used to send and receive messages from the OBD-II port. Section IV is a practical description of the app's operations and layout. This section can be used also as a short starting manual to use the app. Section V describes the architecture in which DRIVES works. In particular, firstly, it describes the in-vehicle elements, then, secondly, the out-of-the-vehicle nodes. Section VI deals with the privacy concerns of our app and infrastructure, underlining the importance of assuring privacy for users' data. Section VII contains the main achieved results by our app and possible future improvements.

II. Related work

As stated in Section I, our app combines two main elements: the collection of vehicle data and the idea to receive personalized services, based on the driving profile. Firstly, we analyze some of the many apps, which recover data from the OBD-II port with an ELM327 dongle. These apps can be downloaded from the Google Play Store for Android or the App Store for iOS. Following, we describe some of these most famous apps.

- *Torque* [3] is probably the most popular Android app with more than 10 million downloads to retrieve real-time data from the OBD-II port. Torque has an intuitive dashboard and it can be used to monitor vehicles' performances.
- *Car Scanner* [4] is a professional car diagnostics solution for Android/iOS. It enables users to customize the layout of the dashboard with gauges and charts. It provides the vehicle's performance data, fault codes and it can also show and reset the Diagnostic Trouble Codes (DTC);
- *OBD Fusion* [5] is an Android/iOS app to create virtual dashboards, showing multiple vehicle sensors, estimate fuel economy, and read DTC. It displays data in a user-friendly view and, in addition to the other apps, it allows users to read and clear also the Check Engine Light (MIL/CEL).

Compared with these apps, our solution has only in common the collection of vehicle data from the OBD-II port using an ELM327 adapter. The target of our app is different, so we have an easier interface, which provides users only information about retrieved data like speed or RPM, and the users can not download data or see any fault code.

Secondly, we describe the driver-rewarding app category, which is composed of apps that provide personalized services to drivers, based on their driving style. Following is a description of some of the main available apps in this category.

- *Connected Cars* [6] is an Android/iOS app that provides drivers a digital connection to their car and get services in return. In particular, the app monitors the car's current health, supplies notifications when the car needs service, and provides the possibility to direct messaging with the maintenance service and booking the repair service. As stated on the app's website, Connected Cars can also analyze the driver's driving behavior, based on the number of hard accelerations, hard braking, and hard turns the driver takes. Besides, this information can be used to get tailored insurance packages based on driving patterns. The app works only with Volkswagen, Audi, Skoda, or SEAT official hardware. To send directly vehicle data to the Connected Cars cloud, it could be necessary to install the OBD unit physical box, which retrieves data from the OBD port; [7].
- *DriveQuant* [8] is an Android/iOS app that analysis the driving style and helps drivers to adopt safer driving behaviors and to reduce fuel consumption. As stated on the download page in Google Play, DriveQuant uses driver smartphone sensors to analyze the trips and calculate driving indicators. This app works only if the driver is a member of a registered company fleet. The registration of the fleet has to be done by contacting DriveQuant company;
- *KnowYourDrive* [9] and *Drivewise* [10] are Android/iOS apps that belong to the set of insurance discount apps. Several insurance companies use a reward system to offer discounts based on a safe driving style. These apps need only to be installed on the smartphone, register the user's data, and the app starts to collect data. Drivers can control their driving style, the achieved discounts, and safe driving tips;
- *DriveScore* [11] is an Android/iOS apps based on the driving style. The app collects driving data from the smartphone and it provides drivers advice to improve their driving skills. It could be a valuable help to analyze driving style and receive insurance discounts.

The main similarity between these apps and DRIVES is that all the apps try to define a driving profile starting from trip or vehicle logs. The previously cited apps like [11] [10] [9] and [8] use smartphone sensors to collect data, while the only app that can use the same system from the OBD-II port like DRIVES is the Connected Cars app. Another difference between our app and the previously described apps is that DRIVES is more flexible because it can add and provide, also after the implementation, different services coming from several providers like charging stations, workshops, restaurants, or markets. Most of the previous apps are focused only on a specific area like insurance, while the first, Connected Cars, which can provide different services like DRIVES, can work only on Volkswagen's group vehicles. Using an OBD-II port dongle, our app can retrieve data from different car brands that implement the standardized OBD-II PIDs (On-Board Diagnostics Parameter IDs), which are codes used to request data from the vehicle, as defined by SAE J1979 [12]. To conclude, to the best of our knowledge, our app can be considered more complete and flexible than the available apps of the driver-rewarding category, because it is designed to work on different car brands and it can provide different services, which can be increased according to the companies joining E-Corridor. Besides, the target of our app is not the sale of products or services, but the promotion of different driving styles.

III. Background

A. Driver DNA

Driver DNA is the metric that we use to define the driver's driving profile. Following this metric, our infrastructure or a service provider can identify a specific driver attitude to be rewarded or discouraged.

Driver DNA was defined in 2017 by the members of the Senseable City Lab of the Massachusetts Institute of Technology (MIT) [2] and applied in several situations like in [13] to define a driver profile. Driver DNA is composed of four parameters (breaking, turning, speeding, and RPM) that can be retrieved from the Controller Area Network (CAN) of

the vehicle and each parameter could represent a particular driving attitude. The vehicles can generate Gigabytes of data per hour [14], so each of the four parameters can be collected during the driving. Following the Driver DNA definition, for a long-period dataset, it is not necessary to store every single value, but one value in a specific temporal window could be sufficient to identify a driving style. For each of the four parameters p, in a specific time or distance window j, all x_p values are averaged together to retrieve a single value $\overline{m}_j$. Later, to have a comparison baseline, the unique score $\overline{m}_j$ is compared with other drivers' values of the same specific road segment, which can be defined as an area with homogeneous driving conditions like, for example, a city center, a flat rural area, or a mountainous area. The dataset of all drivers' values, which, in our solution, is stored in the E-Corridor servers, is divided into quantiles assigning a specific score to each quantile from 0 to 5. With this process, it is possible to define to which quantile each driver belongs and define the driving style with only four values.

The first used parameter is *breaking* which, following its definition in [2], can be retrieved and computed from the frontal acceleration values. The second parameter is *turning* which can be retrieved and computed from the lateral acceleration. The third parameter is *speeding* which can be retrieved and computed from the speed values and its analytic considers also the weather conditions and the speed limit. The fourth parameter is *RPM* which can be retrieved and computed from the RPM values. These four parameters allow defining some driver's attitudes like aggressiveness, comfort, safety, and environmental-friendly driving. Following, in Figure 1, we show how Driver DNA can represent two driving styles using a radar chart, after the computation of each score on the same temporal or distance window, and road segment.

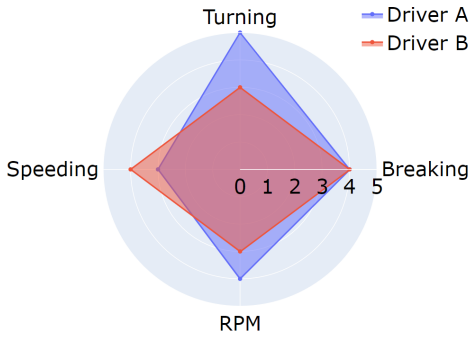

Fig. 1. Radar graph represents two Driver DNAs.

B. E-Corridor

Our app was developed inside the European project E-Corridor, which started in 2020 and will last in 2023, to develop a flexible, secure, and privacy-aware framework aimed at ensuring the safety and security of multimodal transport systems. The project defines a framework architecture, composed of a set of services. For our solution, E-Corridor provides some components to store, analyze, and secure data. In particular, in the architecture, we can find three main different subsystems, namely Information Sharing Infrastructure (ISI), Information Analytics Infrastructure (IAI), and Common Security Infrastructure (CSI). In particular, each subsystem can be defined as:

1) ISI subsystem is *"the process of sharing information in a secure way ensuring data isolation and applying privacy-preserving methods on dataset produced sanitized data"* [15] following a Data Sharing Agreement (DSA);
2) IAI subsystem can *"receives processed information from ISI and extracts intelligence through data mining and information analysis technique"* [15];
3) CSI provides security services like handling cryptographic key generation and secure storage.

These subsystems can be found in each node of the infrastructure and, in particular, our architecture is based on ISI, where data are stored, and on IAI, where data are processed and analyzed.

C. OBD-II Data Collection

Our app collects vehicle data from the OBD-II port using a Bluetooth communication system. For this activity, we leverage on Pires' tools [16]. In particular, for DRIVES, we inherit and apply some concepts and codes of two Pires' works, given under Apache License 2.0: the *obd-java-api* [17] and the *android-obd-reader* [18].

The first is an OBD-II Java API, which provides built-in functions to retrieve data directly from the OBD-II port using a simple string command and a socket. The library is written in Java and it uses OBD-II PIDs to ask data to the port, converting the answers into human-readable results. In our work, we use this library to retrieve some available data from the vehicle like speed or RPM. The second work, the *android-obd-reader*, is an Android app, based on the previous library, which works with Android SDK (API 22, Build tools 23.0.1) or higher. This app shows on the main page several vehicle data and it allows to save and download the trips' logs. From this work, we inherit some graphical solutions and some functions for our app like the settings menu or the OBD II and Bluetooth status, shown on the DRIVES real-data page.

IV. App Design: DRIVES's Front End

DRIVES is designed for Google's Android OS and developed using the official integrated environment Android Studio.

The app can work on every Android device with at least Android 4.0.1 Level 14 version *Ice Cream Sandwich*, released in 2011. In particular, DRIVES is designed to be installed on the infotainment system of the vehicles. For this reason, it is designed to work horizontally with the *Landscape* view and the layout is minimal and clean to allow drivers to select options also during driving, without compromising safety. However, DRIVES can work also with every Android smartphone with the minimum required Android version, so the users can decide to install the app on the smartphone and use it just by activating the Bluetooth and pairing the OBD II dongle.

As shown in Figure 2, DRIVES's homepage enables the selection of the Driver DNA service and it is designed to be user-friendly for any driver.

Fig. 2. DRIVES's homepage. The app design is minimal and clean to allow drivers to select options also during the driving, without compromising safety.

In Figure 3, the second app page allows users to select different options. The first *Services* button serves to ask for a specific service. The second option, *Live Data*, allows the app to show, during the driving, a view with some interesting live vehicle data and it could be used as the driving main screen view.

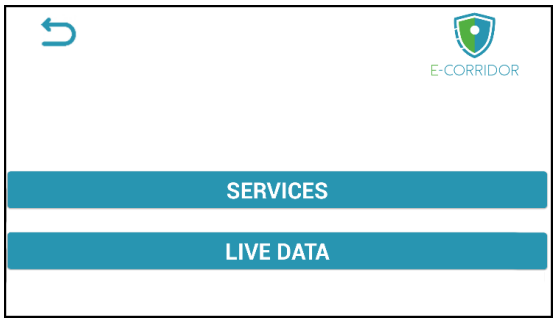

Fig. 3. DRIVES's options page.

The page for the first configuration and, usually, displayed during the driving, is the *Live Data* page. As shown in Figure 4, using the upper horizontal menu during the first configuration, the user can select the *Pair* option and pair the infotainment system or the smartphone with the OBD-II port ELM327 dongle. Then, the user can press the *Settings* button, where different options can be selected:

- *Bluetooth devices*: it enables to select the device from the list of paired devices;
- *OBD protocol*: it enables the selection of the OBD protocol to communicate with the dongle. Usually, with the *auto* option, the app independently finds the working communication protocol;
- *Update periods in seconds*: it defines the update period in seconds of the collected vehicle data and displayed on the screen;
- *Enable full logging*: it is a flag that defines if the app can collect and store vehicle data to compute the Driver DNA or not. If this flag is not selected, the app displays only the live vehicle data without storing it, but, at the same time, it can not update the Driver DNA.

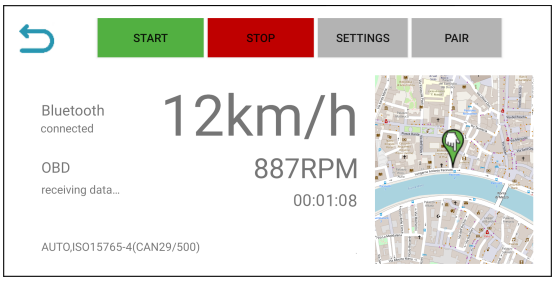

Fig. 4. DRIVES's page which displays real-time data and an OpenStreetMap [19] with a pin on the current vehicle position.

After the first configuration setting, pressing the *Start* button, the user starts the collection of vehicle data. In particular, starting from the left of the screen, DRIVES shows the status of the Bluetooth and OBD connections, and, at the left bottom, the used protocol to communicate with the CAN bus. In the middle of the screen, DRIVES displays the current *speed*, *RPM* and the *engine runtime*, which measures the time since the engine was started and it is shown in the format hours:minutes:seconds. On the right, the app shows an OpenStreetMap [19] with a pin on the current vehicle position.

If the user needs a service, it has to go back to the options view and select *Ask Service*. As shown in Figure 5, there is the possibility to ask for different types of services. In particular, in Figure 5, starting from the left, we show carsharing, a gas/charging station, a restaurant, and a repair workshop.

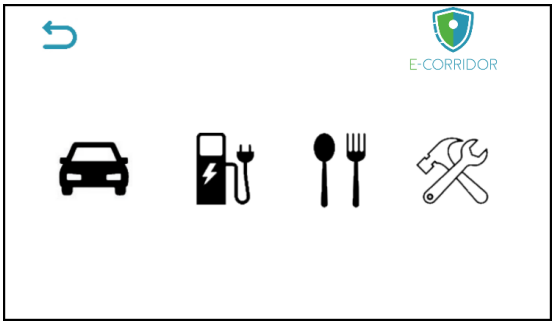

Fig. 5. DRIVES's select services page, which shows four different possible services: a carsharing, a gas/charging station, a restaurant, and a repair workshop.

When the user presses the relative button service, she can select more detailed options for each service type as shown in Figure 6. In this work, we propose the example of a user who is driving a shared car and would like to require a new hourly rental cost, based on her driving. Figure 6 shows the three options. "*My instant price*" option provides a new hourly price, "*Extra discount*" could be a reduction on the actual price, while "*Special gift*" could be another service provided by the rental company.

The car sharing company may incentive users to have an eco-friendly driving style, with, for instance, lower RPM,

Fig. 6. DRIVES's providers page to select different offered services.

offering better hourly rates. When the user presses "*My instant price*" option, the app calls the infrastructure, and the user receives the result of the request as shown in Figure 7. In addition to the result, which is, in this case, an eco-friendly result, the app provides some recommendations on how to achieve a better Driver DNA, and consequently a lower price.

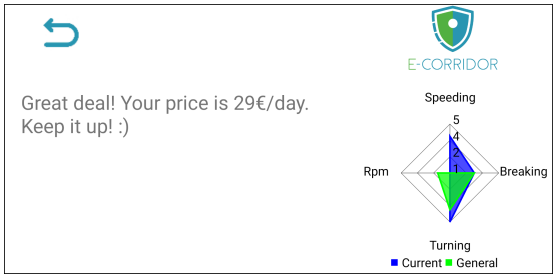

Fig. 7. The result of the user's call for a new rental hourly price.

Figure 7 shows also a radar graph, reporting the instant Driver DNA values and the general Driver DNA as saved in the E-Corridor database.

After the call, the received results are stored in a file on the user's device and in Figure 6 we show the "*My history*" option. Figure 8 shows the history page, where are displayed information on the different user's calls. In particular, it is shown the date-time, the type of required service, the personalized service type, which, in this case, is a rental price, the Driver DNA quartile of the driving style, and the ticket, which is a unique alphanumeric string which identifies the call. This last information could be useful for the user to identify the call when she has to interact with the service provider.

To conclude, a user is able with just a few interactions to receive a service and the solution is scalable because it allows us to add theoretically infinitive services, just depending on the number of providers which will join the E-Corridor project.

V. DRIVES's Back End and Infrastructure

DRIVES works in a communication infrastructure composed of different nodes. In particular, as described in Section III, our main reference is the E-Corridor infrastructure, which allows us to store and analyze data. To describe the DRIVES architecture, we divide the environment into two main areas: the in-vehicle and the out-of-vehicle environment.

Fig. 8. The history page, containing each call information and the unique ticket.

A. In-vehicle

The user can install DRIVES on two different Android device categories: on the vehicle integrated infotainment system or the smartphone. Concerning the operations of DRIVES, the two solutions are equivalent and there are no differences. The only constraints for both solutions are the minimum Android OS version (4.0.1) and the use of the *Landscape* display mode (default for the infotainment system, while in the smartphone the user has just to rotate the screen horizontally). In this section, we will use the generic term of *user's device* to indicate both the possible devices, without defining every time which solution the user has chosen.

The user's device can retrieve the vehicle's data through a Bluetooth connection with an OBD-II port ELM327 dongle. As shown in Figure 9, the OBD-II is a port, usually located under the dashboard, beneath the steering wheel column. Since 1996 the port is mandatory in the USA for all new vehicles and in the following years, almost all countries of the world transposed the rule. The OBD-II is used to access the vehicle's data for various purposes like emissions tests and diagnostics. From this port, it is possible to retrieve different data, sending OBD-II PID. For our app, we decide to use this port to collect data, so, as shown in Figure 9, we use an OBD-II dongle, which is just a Bluetooth adapter to send and receive messages.

Fig. 9. The OBD-II dongle in our test vehicle, located under the dashboard, beneath the steering wheel column.

As shown in Figure 10, the user's device through DRIVES connects via Bluetooth with the dongle and it can start to send and receive messages to be converted to human-readable

vehicle data. During driving, if the user has started the real-time data collection, DRIVES starts to collect data with a frequency defined by the user in the settings options. The default parameter is one data every second, in fact, every second our app sends the dongle the PIDs to require a specific set of parameters, and it receives the relative answers. In particular, DRIVES asks the OBD-II for the following live-data:

- *Speed*: to compute the Driver DNA *speeding* parameter;
- *RPM*: to compute the Driver DNA *RPM* parameter;
- *Engine runtime*: to show on the live-data page the engine runtime;
- *Communication protocol*: to show on the live-data page the used protocol.

At the same time, some other data are collected from the GPS module of the user's device:

- *Latitude*: to compute (combined with longitude) the Driver DNA *breaking* and *turning* parameters;
- *Longitude*: to compute (combined with latitude) the Driver DNA *breaking* and *turning* parameters;
- *Bearing*: to compute the Driver DNA *breaking* and *turning* parameters.

Note that other vehicle parameters could be asked every second, but, when the frequency of request is every one or two seconds and the number of requests becomes significant (more than 15 every time slot), we can notice a slowdown of the answer. This effect can reduce the number of collected data, but it does not affect the Driver DNA, which can be computed even if data have a higher frequency than one or two seconds time slot.

During the driving, DRIVES writes data in a JSON file and stores it on the user's device. The JSON file is structured with several objects with timestamps (e.g. 1638346164442) as keys and its related values. The related values are nested objects with key-value pairs. Following, we report the structure of the values for each timestamp, describing some of the most significant pairs.

- *latitude*: float (e.g. 43.722839);
- *longitude*: float (e.g. 10.403778);
- *bearing*: float (e.g. 289.5);
- *ENGINE_RPM*: string (e.g. 1348RPM);
- *SPEED*: string (e.g. 21km/h);
- *ENGINE_RUNTIME*: string (e.g. 00:22:07);
- *FUEL_LEVEL*: string (e.g. 31.8%);

Then, using the broadband network technology of the user's device, DRIVES sends the JSON file outside of the vehicle and deletes it in the device. This delete operation is performed in only two cases: when the user presses the *Stop* button to stop the real-time data collection or when the number of collected data overcomes 250 instances. An instance is a single set of requests made from the user's device to the OBD-II, containing all the previously required data. The single answer, received from the OBD-II, is written in the JSON file and it is called instance. In a single file, when the number of instances overcomes 250, the JSON file is sent to the E-Corridor infrastructure. The number was arbitrarily chosen after some tests to keep the file size far below one Megabyte to avoid issues or slowdowns during the uploading using broadband network technology.

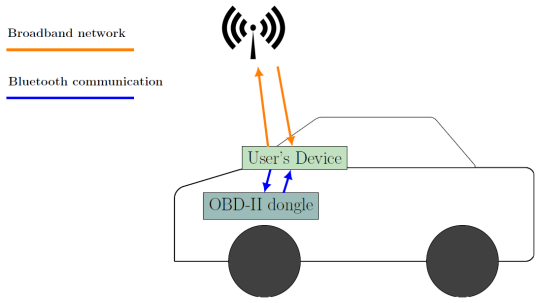

Fig. 10. In-vehicle infrastructure with three main actors (the OBD-II dongle, the user's device, and the RSU) and two connection systems (Bluetooth and broadband network).

B. Out-of-vehicle

The out-of-vehicle infrastructure is represented in Figure 11, where we show the two main communication phases. The first is the *Storing Process* in the upper part of the figure. As already described in the previous subsection, collected data are sent out of the vehicle to be stored in an outer environment. In particular, the app sends out the JSON file with a DSA Id, which is a string to identify the user and its related signed privacy policy. To perform this operation, the ISI infrastructure with its Data Protected Object (DPO), exposes a specific call, `createDPO`, which can be invoked with a POST request on a defined URL. Then, the DPO stores data in specific storage and sends back to the app the string of the result of the POST call with the specific DPO Id to identify the stored JSON file. The *Storing Process* is performed several times during the driving, but this process is completely hidden from the user.

The second event is the *Service Process*, shown in the lower part of Figure 11. During this operation, the users are now active actors because they ask for a specific service and wait for the answer. As described in Section IV, when the users need a service, they have to send the request using the app. The IAI exposes a service called `DriverDNA`, so the app can make a GET call to require the service. Immediately, the IAI, which is the analytic core of the system, starts the Driver DNA computation process, asking the End-Point to retrieve the users' stored data in the ISI. Besides, the End-Point sends a string to inform DRIVES of the start of the analytic. The *Container Driver DNA* (CDD) receives the user's stored data from the storage and it computes for each of the four Driver DNA parameters the average values of the received data to have up-to-date values. Then, the CDD asks the End-Point to retrieve from the ISI the quantile limits to identify in which quantile with respect to the other drivers' database each of the four parameters belongs. The storage contains all the files of the drivers, which use DRIVES, so the IAI will be able to

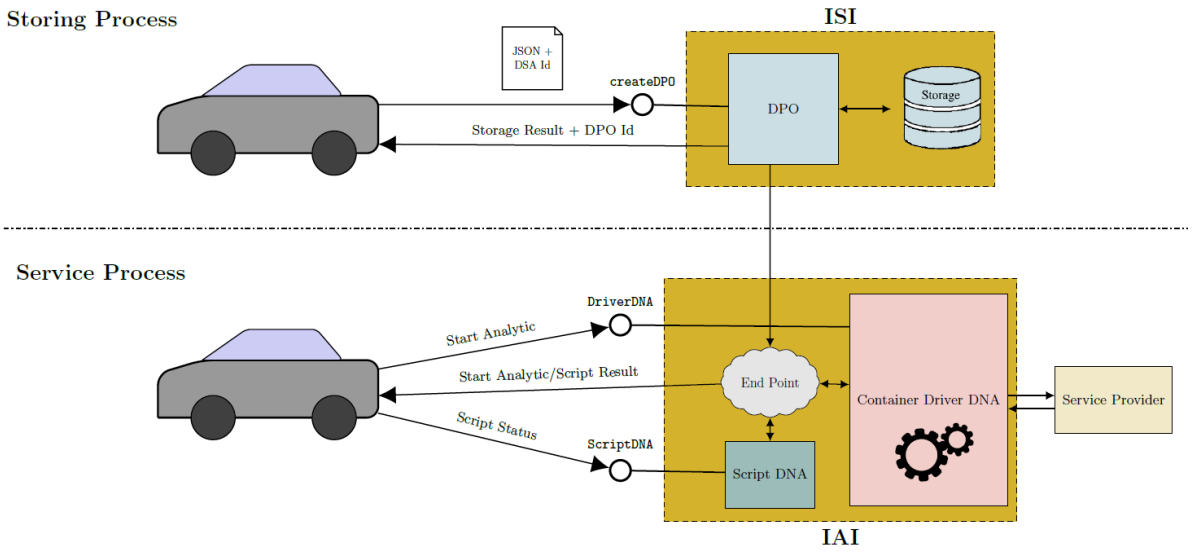

Fig. 11. DRIVES's mixed distributed architecture for storing and service processes. The ISI and the IAI components are part of E-Corridor.

define quantile limits with respect to the other drivers' values. For example, if the CDD has computed a value of 1200 for the RPM and the other drivers' database has the following quantile limits [0-500, 0-999, 1000-1499, 1500-1799, 1800-1999, $>$ 2000], the user's Driver DNA value for RPM will be 2 because it is included in the third quantile limit.

Then, the four Driver DNA values are sent in a string format to the outer Service Provider (SP), without any additional user information. The SP takes the parameters of interest for the required service and it sends the answer to the container. The analytics could take some seconds to be performed and to receive an answer from the Service Provider. For this reason, the infrastructure exposes another call, the `ScriptDNA`, to ask the IAI if the analytic is finished and if it is available the relative result. During this phase, periodically, DRIVES makes a GET call to `ScriptDNA` to receive the result from the IAI. Once the app receives the required result, the process is terminated.

VI. Privacy Concerns

In the last years, the protection of the user's data has become one of the main topics in automotive. Nowadays, modern vehicles can collect and share a significant amount of vehicle data like in our infrastructure. For this reason, it is necessary to study the possible privacy concerns in our app.

To compute the Driver DNA, DRIVES collects several data and it requires some Android authorizations. Figure 12 shows a list of the Android permissions, which DRIVES requires to work. In particular, it requires the permissions to establish a Bluetooth connection (every BLUETOOTH permission), to access the current location (ACCESS_COARSE_LOCATION and ACCESS_FINE_LOCATION), and to write a file in the device's storage (WRITE_EXTERNAL_STORAGE). These permissions are necessary to perform the previous operations and, from Android 10 (API level 29) and higher, the developers must declare these permissions service types. The declaration allows users to express their consent to perform these operations, by clicking on a pop-up multiple-choice window, which appears the first time the user requires a service in which one permission is needed.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.BLUETOOTH_PRIVILEGED"/>
<uses-permission android:name="android.permission.LOCAL_MAC_ADDRESS"/>
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.PARTIAL_WAKE_LOCK" />
<uses-permission android:name="android.permission.READ_LOGS"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
```

Fig. 12. DRIVES's required permissions in the AndroidManifest.xml file.

In this work, to define privacy we use the definition provided by the U.S. National Institute of Standards and Technologies (NIST): "*privacy is the right of a party to maintain control over and confidentiality of information about itself*" [20]. This definition clearly states the right of a party, e.g., a driver, to maintain control over personal information. In our app, the control should be defined in the DSA that a driver signs with E-Corridor. For example, in the European Union, according to Article 12 of the General Data Protection Regulation (GDPR) [21], every DSA should be written "*in a concise, transparent, intelligible, and easily accessible form, using clear and plain language*", allowing a person to maintain full control over personal data. This is fundamental in a privacy-preserving infrastructure like E-Corridor aims to be: it allows users to share personal data for different purposes, but also it requires giving the driver full control over them.

The other part of the definition requires data confidentiality. We can again refer to the NIST definition of *confidentiality*: "*the property that data or information is not made available or disclosed to unauthorized persons or processes*" [22]. This property may be assured in the E-Corridor infrastructure, using encryption methods like symmetric or homomorphic encryption for the storage of the user's data. As shown in Section V-B user's data remains inside the E-Corridor infrastructure where the DSA and the Android DRIVES Manifest give users control over their data and confidentiality is assured by the encryption methods. Data are sent out of the infrastructure, only when are sent to the Service Provider. As stated in Section V-B, the only data that the Service Provider receives are the Driver DNA values, which do not contain any reference to the user's personal information. Besides, the only Driver DNA values do not allow a Service Provider to identify or profile a specific user, because there is not element to distinguish values from one user to another user. To conclude, following the NIST privacy definition, we can state that DRIVES in the E-Corridor infrastructure can preserve the privacy of users' data, give them full control over data, and without disclosing any information.

VII. Conclusion and future work

In this work, we have described our Android app, DRIVES, which can provide users with customized services based on their driving style. Besides, we show the feasibility and the opportunities that DRIVES can disclose. In particular, our app can encourage safer and more sustainable mobility, giving incentives like a price discount to virtuous drivers. For these reasons, DRIVES may contribute to moving to a safer and more sustainable road ecosystem.

At the same time, we think that our app can be expanded, for example, by studying the application on autonomous vehicles, where the driving is performed by Artificial Intelligence. A deeper study on this topic can introduce a driving style profile also for autonomous drivers like an *Artificial Intelligence Driver DNA* to encourage developers to create a driving profile that can be rewarded by an app. We think that another future work could be the study of a more efficient storing process for the storage, because, on a large and commercial application of DRIVES, it is not feasible to have only one place to store driver's data. One solution could be to have multiple storages in different locations and compute the user's Driver DNA related only to the specific location where the user is driving.

To conclude, we can state that DRIVES could be an example and a starting point to study and increase the driver-rewarding app category for vehicles and Android/iOS devices.

Acknowledgment

The project leading to this application has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 883135 (E-Corridor).

References

[1] Y. Lei, K. Liu, Y. Fu, X. Li, Z. Liu, and S. Sun, "Research on driving style recognition method based on drivers dynamic demand," *Advances in Mechanical Engineering*, vol. 8, 09 2016.

[2] U. Fugiglando, P. Santi, S. Milardo, K. Abida, and C. Ratti, "Characterizing the "driver dna" through can bus data analysis," in *Proceedings of the 2nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services*, ser. CarSys '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 37–41. [Online]. Available: https://doi.org/10.1145/3131944.3133939

[3] I. Hawkins. Torque. Accessed on July 12, 2022. [Online]. Available: https://torque-bhp.com/

[4] 0vZ. Car scanner. Accessed on July 12, 2022. [Online]. Available: https://www.carscanner.info/

[5] OCTech. Obd fusion. Accessed on July 12, 2022. [Online]. Available: https://www.obdsoftware.net/software/obdfusion

[6] Connected Cars. Accessed on July 14, 2022. [Online]. Available: https://connectedcars.io/

[7] Connected Cars. Accessed on July 14, 2022. [Online]. Available: https://connectedcars.io/device/

[8] Drive Quant. Accessed on July 14, 2022. [Online]. Available: https://www.drivequant.com/

[9] American Family Insurance. Knowyourdrive. Accessed on July 14, 2022. [Online]. Available: https://play.google.com/store/apps/details?id=com.amfam.ubi&hl=en&gl=US

[10] Allstate Insurance Co. Drivewise. Accessed on July 13, 2022. [Online]. Available: https://www.allstate.ca/webpages/auto-insurance/drivewise-app.aspx

[11] ClearScore. Drivescore. Accessed on July 15, 2022. [Online]. Available: https://www.drivescore.com/

[12] SAE. (2017) Sae j1979_201702. Accessed on July 9, 2022. [Online]. Available: https://www.sae.org/standards/content/j1979_201702

[13] G. Costantino, F. Martinelli, I. Matteucci, and P. Santi, "A privacy-preserving infrastructure for driver's reputation aware automotive services," in *Socio-Technical Aspects in Security and Trust - 9th International Workshop, STAST 2019, Luxembourg City, Luxembourg, September 26, 2019, Revised Selected Papers*, ser. Lecture Notes in Computer Science, vol. 11739. Springer, 2019, pp. 159–174.

[14] E. Massaro, C. Ahn, C. Ratti, P. Santi, R. Stahlmann, A. Lamprecht, M. Roehder, and M. Huber, "The car as an ambient sensing platform," *Proc. IEEE*, vol. 105, no. 1, pp. 3–7, 2017. [Online]. Available: https://doi.org/10.1109/JPROC.2016.2634938

[15] F. Martinelli, A. Saracino, and M. Sheikhalishahi, "Modeling privacy aware information sharing systems: A formal and general approach," in *2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, August 23-26, 2016*. IEEE, 2016, pp. 767–774. [Online]. Available: https://doi.org/10.1109/TrustCom.2016.0137

[16] Pires, "Project title," https://github.com/pires, accessed on July 10, 2022.

[17] Pires, "obd-java-api," https://github.com/pires/obd-java-api, accessed on July 10, 2022.

[18] Pires, "android-obd-reader," https://github.com/pires/android-obd-reader, accessed on July 15, 2022.

[19] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org," https://www.openstreetmap.org, accessed on July 10, 2022.

[20] A. Oldehoeft, "Foundations of a security policy for use of the national research and educational network," 1992-02-01 1992.

[21] European Parliament and Council of the European Union, "Eu general data protection regulation (gdpr): Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016." 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj

[22] M. Scholl, K. Stine, J. Hash, P. Bowen, L. Johnson, C. Dancy, and D. Steinberg, "An introductory resource guide for implementing the health insurance portability and accountability act (hipaa) security rule," 2008-10-22 04:10:00 2008. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=890098