Telematics-enabled Remote Diagnosis Solution

Overview

Telematics helps to improve after-sales service by providing manufacturers with real-time access to vehicle data. Telematics enables transmission of information from an automobile's electronic control unit to a data center. The Remote Diagnosis Application is a diagnostic tool developed for telematics-enabled vehicles. Technicians use the diagnosis application to check the overall health of the vehicle. Data analysis of snapshots helps technicians to identify and fix faults. The application receives data from a host of services such as the vibration analysis application (uses in-built sensors to detect fault sources), accelerometer, gyrometer, GPS, and street drive simulations.

The Remote Diagnosis Application radically simplifies the inspection and diagnosis process by providing technicians with an insightful pocket diagnostic tool.

Client Profile

Based in Europe, our client is the research and development center for one of the world's largest manufacturers of premium and commercial vehicles. The center focuses on research, IT engineering, and product development.

Business Case

Previously, each workshop was assigned a diagnostic kit, a large wired device that was connected to the vehicle for scanning and recording vehicle data. The client required a mobile application that would replace this system and enable technicians to diagnose and analyze faults in cars and trucks remotely.

AUTOMOTIVE TELEMATICS MARKET TO REACH USD 415.93 BILLION BY 2027 CAGR OF 26.4%: EMERGEN RESEARCH

Solution

The Remote Diagnosis Application assesses diagnostic data of telematics-enabled vehicles to help technicians in remotely identifying issues. Technicians scan the vehicle identification number and check for faults recorded on the electronic control unit.

Features required for diagnosis such as circuit diagrams, machine parts required for fixing faults, and possible causes are made available for an easy and quick diagnosis. The solution comprises the following components:

- PWA App
- Nginx
- SSO module
- Microgateway
- Microservices

Prior to our involvement, the client had a prototype of the Remote Diagnosis Application which was developed based on a monolithic architecture. Deployment was a huge task that involved uploading 2.5 GB of the build package. By moving to microservices architecture, QBurst was able to deliver a flexible and scalable alternative that was much faster to deploy.

The Remote Diagnosis Application was also made available to Independent Service Providers (ISPs) in compliance with EU regulations. Ten APIs were developed for ISPs, six for the remote diagnostic system, and four for the remote maintenance system. Using these APIs, vehicles can be diagnosed remotely as well as from the service center.

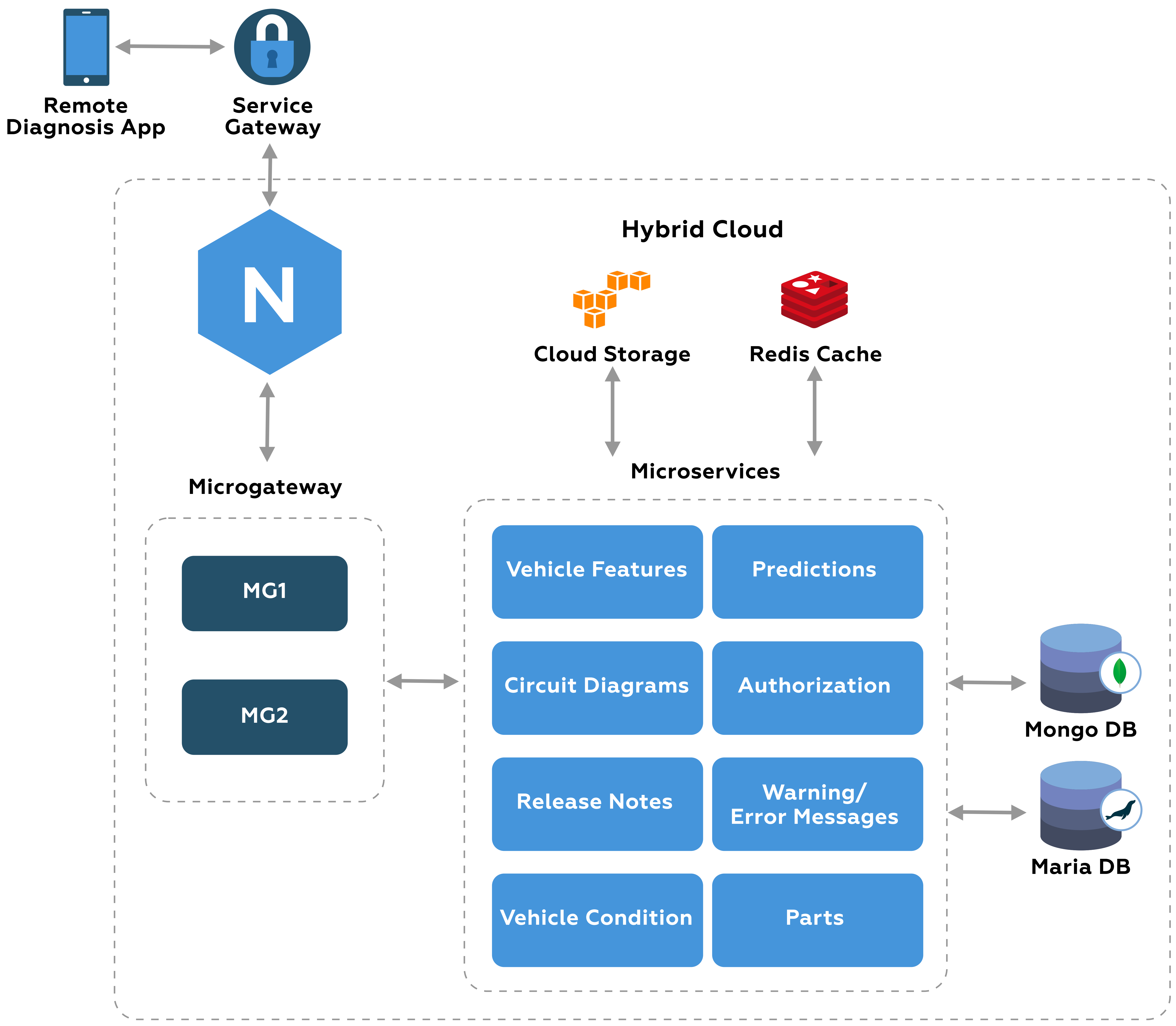
- The microservices-based solution was deployed on the cloud by integrating functionalities such as auto-scaling, auto-deployment, quality gate analysis, and health alerts.
- While the app integrates with multiple third-party services that have their own quota and spike arrest, data caching helps avoid repeated API calls ensuring faster response time.
- To ensure support for thousands of users across the globe, the application was load tested at various ranges.
- Leveraging Kubernetes' extended features for Horizontal Pod Autoscaling (HPA), we provided an auto-scalable backend.
- HPA, PDB (Pods Distribution Budget), and rolling update strategies helped achieve zero downtime during deployment.
- Deployments are triggered automatically in respective environments through Jenkins jobs with minimal developer effort.
- Each of the multiple deployment lines for development, staging, UAT, and production require peer review and confirmation before being pushed through.
- Grafana and Prometheus are integrated in the backend for real-time monitoring and visualization.
- Services such as notifications and documentation (audit protocol) require real-time consolidation of data across numerous third-party services.
- Error handling and thread-based approach ensure wait time per user is minimal.

Key Challenges

- One of the challenges of moving to a microservices-based architecture is that there are many components that need testing. QBurst played a crucial role in identifying issues during migration. Initially we released a minimal set of features focusing primarily on quick-tests. All diagnoses were recorded and stored on the device. However, the cost involved in clearing the native applications prior to each release was a challenge. We were able to completely eliminate this problem with a PWA version developed with new features.

- Another challenge was storage and sharing of vehicle data with multiple users within the workshop. Data was stored on the device with no link to the workshop or technicians. We effected extensive architectural changes to manage data inputs from multiple users assigned to one vehicle. This prevented mismatch of data when multiple users simultaneously worked on a vehicle.

- Concerns raised by customers were not recorded in the app and diagnosis was purely based on fault codes of the quick-test. This led to another significant change in the architecture. We implemented a concern-based approach wherein most features are linked to customer concerns. Following this, the entire diagnosis was based on the premise of concerns raised by customers.



Key Features

The application begins a diagnostic session either by manual entry of the vehicle number or by scanning the vehicle ID. Scanning is a core feature of the Remote Diagnosis Application.

- **Quick-test:** Analyzes the vehicle's current state by viewing the control unit and associated fault code of the vehicle

- **Guided test:** Based on this analysis, a guided test provides a list of parts that can be used to fix issues associated with the fault code

- **Repair forecast:** Generates predictions based on big data analysis of the fault codes associated with similar cases; presents a list of parts, labor involved, and recommendations to resolve faults

- **Essential parts:** Technicians can add parts that can be used to resolve faults in the vehicle, which can be shared via email to supervisors to initiate purchase orders
- **Historical data:** The historical data associated with fault occurrence helps identify repeating faults
- **Messages:** Warnings to display any active errors in the vehicle
- **Vehicle build info:** An overview of make and model
- **Reset fault code:** Clears all the current faults in a vehicle and triggers a new quick-test to identify remaining issues
- **Vehicle health and maintenance:** Displays information related to service due date, mileage, and on-the-road data
- **Repair forecast:** Consumes an API for predictive analysis and uses feedback mechanism to improve prediction accuracy
- **Diagnostic capability:** Meant for vehicles that support telediagnosis and provides an overview of capabilities of the current vehicle
- **Handover:** Cases that cannot be diagnosed through the app can be handed over to the diagnosis team
- **Link to client portal:** Records of vehicles brought in for service are pushed to the portal; one-way sync of data enables technicians to start with diagnosis right away instead of manually entering customer concerns into the app
- **Recording symptoms:** A symptom tree represents the classification of an issue in the vehicle (for example, body > chassis > door > rear > left > doorknob), and possible solutions are retrieved
- **Tips:** Tips based on symptoms documented for an issue help in quick resolution of the issue
- **Documentation/audit-protocol:** Technicians can document diagnostic findings in detail when cases are closed, which is then displayed under appropriate sections in a PDF document, giving workshop supervisors a clear idea of work done
- **Media protocol:** Technicians add annotated images and diagrams that enrich the documentation
- **Symptom software updates:** Lists available software updates for electronic control units based on manufacturer code and symptoms

Highlights

- Compatible with iOS, Android, and PWA platforms
- CI/CD pipeline for deployment in dev environments
- Zero downtime during deployment achieved through an orchestrated pipelining of Sonar, Jenkins, Docker, and K8s
- Microservices to deploy features ensures flexibility and ease of maintenance
- Authentication and authorization at every layer ensures data protection
- Adobe Analytics and AppDynamics for analytics

- Generates PDF report of the diagnostic session
- 23 languages supported
- Grafana and Prometheus for analysis of different tenants
- ELK stack for analysis of gateway and database logs

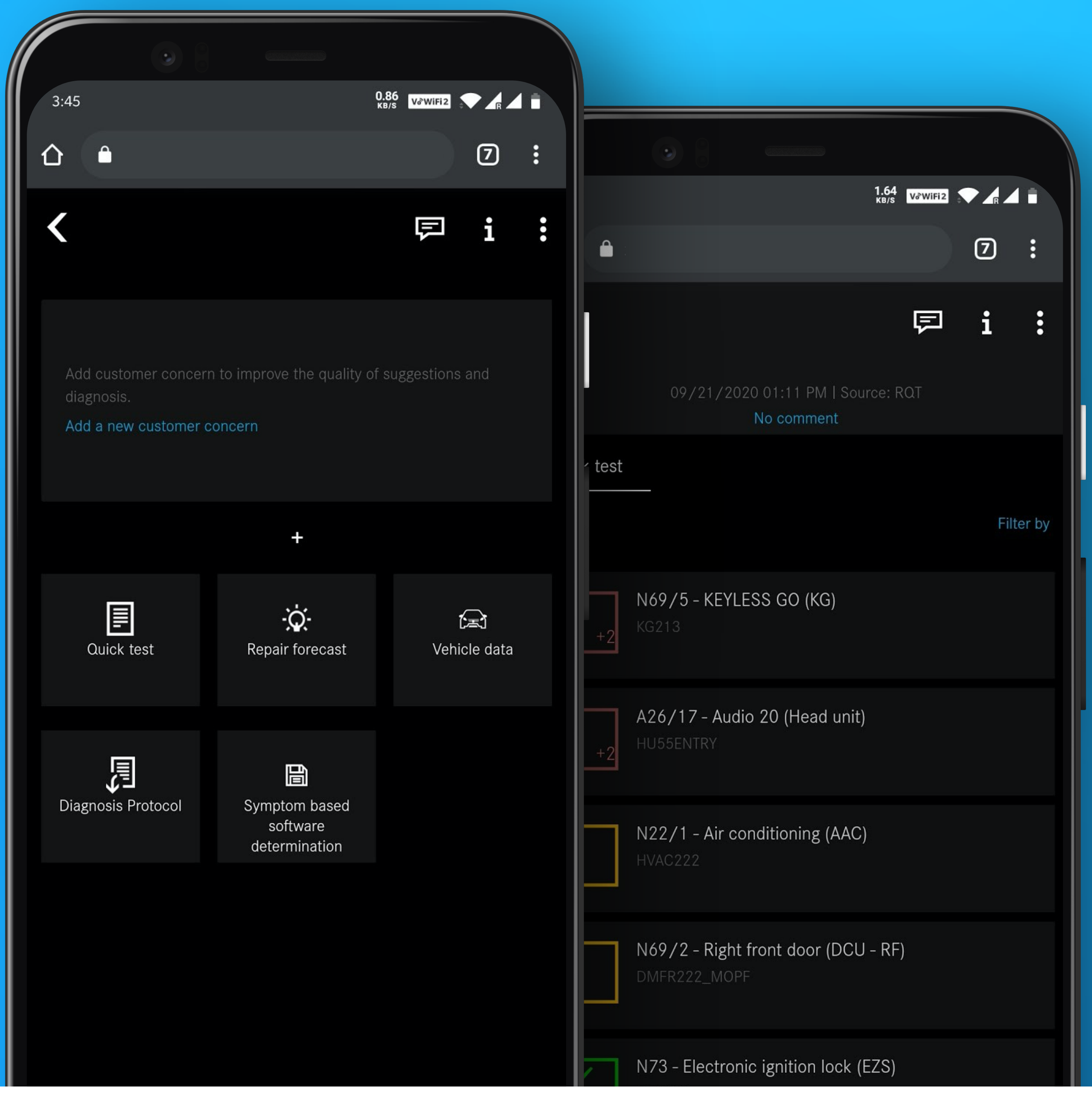
Technologies

- Java
- iOS, Android
- PWA
- Angular
- Adobe Analytics
- AppDynamics
- Tesseract
- ELK stack
- Grafana, Prometheus
- Kubernetes
- Docker
- Jenkins
- MariaDB
- MongoDB

APIs: Spring boot, OpenID Connect, Apigee Microgateway, Kubernetes, Jenkins, Git Quay, DHC SonarQube, PostgreSQL, Docker, Filebeat

Business Benefits

- The application was released to market as scheduled and was well received by workshop technicians because of its usability and ease of access
- Achieved radical simplification in analysis and diagnosis of vehicular faults
- Improved customer satisfaction with remote support and reduced maintenance costs as a result of preventive diagnostics
- Increased productivity enabling technicians to diagnose vehicles on their smartphones
- Replaced bulky diagnosis kit, availability of which was limited to one per workshop, with a pocket-friendly tool
- PWA version compatible with all smartphones
- Our effort to identify, document, and devise solutions (in the design phase) to mitigate scenarios related to concurrent usage was appreciated by the client


Add customer concern to improve the quality of suggestions and diagnosis.

09/21/2020 01:11 PM | Source: RQT

N69/5 - KEYLESS GO (KG)
KG213

A26/17 - Audio 20 (Head unit)
HU55ENTRY

N22/1 - Air conditioning (AAC)
HVAC222

N69/2 - Right front door (DCU - RF)
DMFR222_MOPF

N73 - Electronic ignition lock (EZS)