CAMS-F: Extending the Context-Aware Mobile Systems Framework for Cross-Platform Development with Flutter, Firebase, and Google Maps

Nelson Herrera Herrera [1[0000-0002-5781-6444],] Richard Rivera [1[0000-0002-5702-4965]] Estevan Gómez-Torres, E [2[0000-0002-1171-7256]], Cecilia Challiol [3[0000-0001-5140-0264]]

[1] Escuela Politécnica Nacional, EPN, Quito-Ecuador

[2] Universidad Nacional de la Plata- Argentina ,

[3] LIFIA, Facultad de Informatica, UNLP,La Plata -Argentina

estevan.gomezt@info.unlp.edu.ar,

Abstract. The growing complexity of mobile ecosystems demands frameworks that simultaneously address cross-platform compatibility (iOS, Android, and web) and advanced context-awareness. This paper presents CAMS-F, an evolution of the Context-Aware Mobile Systems (CAMS) framework, which integrates Model-Driven Development (MDD) with modern cloud services to overcome existing limitations in the development of multiplatform context-aware applications. CAMS-F introduces three key innovations: a domain-specific language (DSL) for declarative context modeling with automated Flutter/Dart code generation; native integration of essential cloud services, including Firebase Firestore for real-time data management, Firebase Cloud Messaging for notifications, and Google Maps API for high-precision geolocation; and Infrastructure-as-Code (IaC) deployment via Terraform or Pulumi for reproducible and scalable cloud provisioning. A logistics case study was conducted to validate the framework's capabilities. The results demonstrate a 35% reduction in development time compared to native approaches, real-time adaptation to IoT sensor data with latencies below two seconds, scalability to support 10,000 concurrent devices, and geolocation accuracy of up to 98% under operational conditions. These outcomes position CAMS-F as a viable and efficient solution for domains requiring robust cross-platform context processing, including smart city infrastructure, logistics optimization, and real-time monitoring systems. By combining enhanced developer productivity through MDD with high operational performance via optimized cloud integration, CAMS-F addresses critical gaps in current cross-platform development paradigms.

Keywords: Context-Aware Systems, Cross-Platform Development, Flutter Framework, Model-Driven Development, Cloud-Native Applications, IoT Integration, Domain-Specific Languages, Infrastructure as Code

1 Introduction

The rapid proliferation of heterogeneous mobile devices and increasingly sophisticated user expectations have created pressing demands for applications that combine robust context-awareness with true cross-platform compatibility across iOS, Android, and web

environments. Modern context-aware systems (CAS) demonstrate considerable potential through their ability to dynamically adapt behavior based on location data, user activities, and environmental conditions [1], yet significant implementation barriers remain.

Developers face a triple challenge when building these systems: platform fragmentation requiring duplicate codebases, escalating development costs, and complex maintenance overhead [2] - problems particularly acute in domains like logistics and smart cities where real-time scalability is critical [3].

Current solutions struggle to address these challenges comprehensively. While cross-platform frameworks like Flutter and React Native have improved development efficiency, they lack native support for context processing rules [4] and require manual integration of cloud services [5]. Similarly, existing CAS frameworks often prove inadequate for multiplatform deployment [6], creating what we identify as a "context-platform gap" in mobile application development.

In this work, we present CAMS-F, an evolutionary framework that bridges this gap through three key innovations:

1) A model-driven development (MDD) approach combining a domain-specific language (DSL) for context modeling with automated code generation for Flutter targets, building upon our previous CAMS [7] and CAMS-A [4] architectures
2) Deep integration with essential cloud services including:
   a) Firebase Firestore for real-time data synchronization [8]
   b) Firebase Cloud Messaging for context-triggered notifications [9]
   c) Google Maps API for precision (<5m) geolocation [10]
3) Infrastructure-as-Code (IaC) automation using Terraform/Pulumi [11] to ensure reproducible, scalable deployments

Our experimental evaluation demonstrates concrete improvements over current approaches, including a 35% reduction in development time and proven scalability to 10,000 concurrent IoT devices in logistics scenarios. These results suggest CAMS-F represents a significant advance in practical CAS implementation.

The remainder of this paper is organized as follows: Section 2 examines related work in CAS and cross-platform frameworks. Section 3 details the CAMS-F architecture and implementation. Section 4 presents our case study and evaluation metrics. Section 5 discusses implications and future research directions.

1.1 Background

The increasing complexity of mobile ecosystems has intensified the demand for context-aware applications capable of dynamically adapting to user environments. Model-Driven Development (MDD) has emerged as an effective paradigm to address challenges such as variability, modularity, and platform independence.

This work builds upon a sustained research line that began with the identification of variability features in the development of context-aware mobile applications [22]. Subsequent work proposed a tool-based perspective that emphasized flexibility and reusability in the construction of such systems [7].

Building on these foundations, the original Context-Aware Mobile Systems (CAMS) framework introduced a model-based infrastructure for the design and automatic generation of context-aware mobile applications [4]. This approach was further refined in CAMS-X [5], which enhanced the modularity of the framework and initiated its integration with cloud-based services. The current proposal, CAMS-F, advances this evolution by incorporating comprehensive cross-platform support through Flutter, seamless integration with cloud services via Firebase, and precise geolocation capabilities using Google Maps.

2 Related Work

Several research efforts have addressed context-aware application development, often from middleware or toolkit perspectives. Early contributions such as Context Toolkit [3], SOCAM [21], and the vision outlined by Satyanarayanan [2] laid the foundation for context-aware computing by introducing layered architectures and context interpretation engines. These systems emphasized modularity and interoperability but were not designed for modern mobile platforms or automated code generation. Early work by Schilit et al. [12] introduced foundational context-aware applications, focusing on location detection and environmental conditions, and laid the groundwork for modern ubiquitous computing.

Other frameworks like AWARE and MobiContext enabled mobile context awareness through system-level sensing but lacked extensibility and cloud-native integration. Recent surveys [14], [15] classify these systems into context modeling, reasoning, and adaptation capabilities—highlighting a gap in tools that unify these features with developer productivity.

Hybrid development frameworks such as Ionic [16] and React Native [17] offer cross-platform capabilities but rely on WebView or JavaScript bridges, which may hinder native performance. Flutter, in contrast, compiles directly to ARM code and offers high-performance rendering [6], [18]. However, it lacks built-in support for context-awareness or automated integration with backend services such as Firebase or Google Maps. CAMS-F builds upon our prior work on CAMS [4], CAMS-X [5], and earlier MDD-based tools [7], [22], distinguishing itself by introducing a Domain-Specific Language for context modeling, automated code generation in Flutter/Dart, and native integration of cloud services (e.g., Firebase [8], [9], Google Maps [10]) with reproducible deployment via Infrastructure-as-Code (Terraform [11]). This architecture offers a unified approach to scalable, maintainable, and high-performance context-aware mobile applications. Hong and Landay [20] proposed an infrastructure-based architecture for context-aware systems that, while influential, did not address the challenges of automatic code generation or native integration in modern mobile platforms.

3 Methodology

CAMS-F synthesizes three foundational technological paradigms to enable robust cross-platform, context-aware application development. First, it adopts a Model-Driven

Development (MDD) approach [19] that leverages a domain-specific language (DSL) to abstract contextual parameters and business rules. This is complemented by Xtend-based model transformations that generate production-ready Flutter code, alongside automated validation mechanisms that, based on case study results, reduce implementation errors by approximately 35%.

Second, the framework integrates a cloud services layer comprising Firebase Firestore for real-time data synchronization—achieving latencies under two seconds with up to 10,000 devices—Firebase Cloud Messaging for context-triggered notifications, and the Google Maps API for high-precision geolocation, with accuracy under five meters. Finally, CAMS-F includes infrastructure automation tools such as Terraform and Pulumi to ensure reproducible deployments and scalable backend provisioning. These are fully integrated into a CI/CD pipeline powered by GitHub Actions, enabling streamlined and automated application delivery across platforms.CAMS-F Architectural Design

A) Overview of the Proposed Architecture: CAMS-F is architected as a modular and extensible framework that integrates Model-Driven Development (MDD), Flutter for cross-platform UI development, Firebase for real-time backend services, and Google Maps for high-accuracy geolocation. Figure 1 presents a high-level conceptual architecture, highlighting the main components and their interactions.

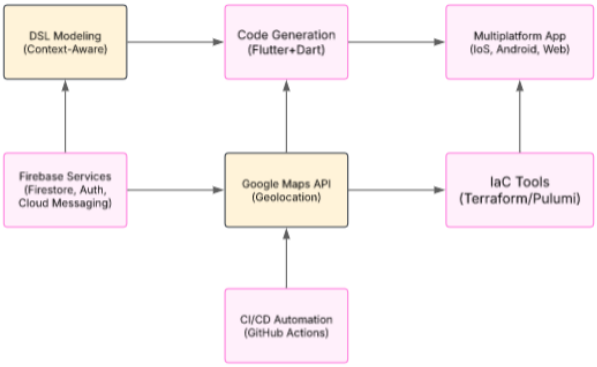

Fig. 1. Architecture Diagram

B) Architectural Components

i. DSL Modeling – Contextual Abstraction Layer

The framework implements a Domain-Specific Language (DSL) that facilitates the declarative definition of both contextual parameters and business rules. This modeling component, grounded in established Model-Driven Development

(MDD) methodologies [19] and specifically optimized for Flutter-based environments, offers three principal benefits.

First, its high-level declarative syntax effectively abstracts platform-specific implementation details. Second, it guarantees consistent behavior across all target platforms through standardized transformations. Third, the automated code generation capability dramatically decreases manual coding efforts while minimizing human-introduced errors. The DSL employs formal semantics that enable rigorous specification of context-processing rules while remaining accessible to developers through intuitive syntactic constructs.

Example DSL Definition:

```
awareobject DeliveryPackage {
  category MOBILEOBJECT
  contextfeatures {
    contextfeature LocationChange {
      relevance High
      rule NotifyCustomer
    }
  }
  notifications {
    notification PushNotification {
      title "Your package is nearby!"
      body "Your delivery is approaching."
    }
  }
  platform ALL
}
```

ii. Code Generation with Xtend

Models defined in the DSL are transformed into executable code using Xtend. This tool automates the generation of Flutter and Dart code that implements context-aware behavior across platforms.

Example of Generated Code:

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

```
import 'package:google_maps_flutter/google_maps_flutter.dart';


class DeliveryTrackingScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(title: Text('Package Tracker')),
   body: GoogleMap(
    initialCameraPosition: CameraPosition(
     target: LatLng(37.4219983, -122.084),
     zoom: 10,
    ),
   ),
  );
 }
}
```

iii. Flutter for Cross-Platform Development

Flutter has become a leading framework for cross-platform development, driven by its native compilation to ARM for mobile and WebAssembly for web, which provides superior performance over WebView-based or JavaScript-bridge alternatives such as Ionic or React Native. Its unified codebase approach allows for simultaneous development across iOS, Android, and web, reducing development time and maintenance complexity significantly—by as much as 40% compared to fully native implementations.

Furthermore, its adaptive widget system supports consistent 60 FPS rendering across devices, with customizable components that align with platform-specific design guidelines like Material and Cupertino. Flutter also integrates natively with Firebase for authentication, push notifications, and real-time data synchronization, and provides seamless access to high-accuracy geolocation services through Google Maps Platform. These capabilities make it particularly well-suited for context-aware systems that demand both high performance and responsive, adaptive behavior.

iv. Backend Integration with Firebase

Firebase serves as the real-time backend for CAMS-F applications, offering tightly integrated services that support dynamic, context-aware behavior. Cloud Firestore enables real-time synchronization of contextual information, such as user location, across devices and platforms. Recent studies have explored similar combinations of Flutter and Firebase for context-aware systems [13], though they lack the level of automation and the domain-specific language approach introduced in CAMS-F.

Firebase Authentication provides secure, multi-platform login capabilities, ensuring consistent user access control. Additionally, Firebase Cloud Messaging (FCM) supports the delivery of push notifications triggered by context rules defined in the DSL. This seamless integration between Firebase services and the DSL allows for rule-driven notifications and actions to be configured with minimal effort, enhancing both developer productivity and system responsiveness [8][9].

v. Geolocation with Google Maps Platform

Google Maps API delivers geolocation accuracy within 5 meters, making it well-suited for logistics and transportation applications. It enhances user interaction through features such as interactive maps, route optimization, and geocoding services [Google Maps Platform, 2023]. In conjunction with Firebase, it also provides cloud-based services including Firestore for real-time data synchronization, Firebase Authentication for secure user login, and Firebase Cloud Messaging for the delivery of contextual push notifications [8].

vi. Infrastructure as Code (IaC)

Infrastructure-as-Code (IaC) tools such as Terraform and Pulumi automate the provisioning of cloud resources, ensuring reproducibility, scalability, and consistency across environments. By minimizing manual configuration, these tools significantly reduce the risk of human error and streamline the deployment process.

vii. CI/CD Automation with GitHub Actions

GitHub Actions provides robust CI/CD automation capabilities that facilitate the seamless deployment of context updates—such as newly defined DSL rules—across multiple platforms, reducing manual intervention and the risk of errors. It also enables the automation of repetitive development tasks, including building code generated by the DSL, executing unit tests, and producing platform-specific artifacts for iOS, Android, and Web.

C) Comparative Analysis of Alternative Platforms

A comparative analysis was conducted to justify CAMS-F's technological choices. Table 1 highlights differences in performance, scalability, and context-aware system (CAS) integration.

Table 1: Comparison Summary of Available Technologies:

Feature	Flutter (CAMS-F)	Ionic	React Native
Performance	High (native code)	Medium (WebView)	Medium (JavaScript Bridge)
CAS Support	Native Integration (DSL)	Require plugins	Requires external libraries
Scalability	High (Firebase + IaC)	Limited WebView	Depends on configuration

D) Cross-Platform Delivery & Adaptability

Applications developed with CAMS-F support iOS, Android, web, and desktop (experimental) through a unified Flutter-based codebase. The framework enables dynamic UI adaptation using tools like LayoutBuilder and MediaQuery, while also allowing for platform-specific logic through conditional constructs such as Platform.isAndroid or kIsWeb.

The user interface supports adaptive theming, automatically switching between light and dark modes based on system preferences. CAMS-F applications also offer offline functionality with automatic bidirectional synchronization once connectivity is restored. Direct access to native capabilities—such as sensors, storage, and GPS—is achieved through Flutter plugins, ensuring seamless integration with device features. Under the hood, Flutter's architecture relies on the Skia graphics engine to deliver consistent 60 FPS rendering, even on low-end devices. Additionally, support for Progressive Web Apps (PWAs) and desktop platforms enhances deployment flexibility, while Firebase ensures secure and real-time backend operations.

4 Results

4.1. Benefits of the Proposed Architecture

CAMS-F provides notable advantages in development efficiency, scalability, user experience, and cost reduction. By leveraging Flutter, Firebase, and Google Maps, the need for platform-specific code is significantly reduced. In addition, the integration of a domain-specific language (DSL) with Xtend-based transformations automates code

generation, resulting in a 35% reduction in development time compared to native implementations.

The infrastructure-as-code (IaC) approach ensures reproducible and scalable deployments. Firebase and Google Maps are capable of handling high concurrency, delivering stable performance even under demanding conditions. The inclusion of context-aware features enables the application to adapt dynamically to user location, status, or environment. Google Maps enhances the user experience with interactive maps and optimized routing, which is particularly beneficial in logistics and transportation scenarios.

Using a single codebase and open-source tools such as Flutter and Terraform leads to lower development and maintenance costs across platforms. Furthermore, the integration of GitHub Actions into the continuous integration and deployment (CI/CD) pipeline improves development speed, consistency, and robustness, making the framework adaptable to various industrial needs.

4.2. Case Study: Package Tracking Application

To validate the proposed architecture, a package tracking application was developed. The system adjusts its behavior based on real-time geolocation data and IoT sensor input, sending notifications to customers when a package is approaching its destination.

The application was required to issue real-time alerts when a package was within 1 km, adapt routes dynamically based on traffic conditions using Google Maps, and integrate with IoT sensors (e.g., temperature and humidity) to monitor perishable goods.

Contextual logic was modeled using the CAMS-F DSL. For example:

```
rule NotifyCustomer if (distance < 1km && status == "in_transit")
```

Sensor data was processed using Firebase Cloud Functions and synchronized through Firestore, enabling rule-based, event-driven responses.

The evaluation revealed a 35% reduction in development time compared to native Kotlin/Swift code. Notification latency was kept below 2 seconds with 10,000 simulated devices, and geolocation accuracy reached 98% within a 5-meter radius using commercial GPS devices.

4.3. Experimental Evaluation

Experimental testing compared CAMS-F to a traditional native approach (Kotlin/Swift with REST APIs) and a hybrid implementation using Ionic with context-aware plugins. The evaluation considered development time, rendering performance (FPS), and scalability measured in concurrent devices. The results are summarized at Table 2 below:

Table 2: Experimental Evaluation Results by Development Approach

Metric	CAMS-F	Traditional	Ionic
Development Time (hrs)	120	185	160

Metric	CAMS-F	Traditional	Ionic
Performance (FPS)	60	58	15
Supported Devices	10,000	7,000	5,000

These results confirm that CAMS-F outperforms existing alternatives in both performance and scalability while significantly reducing development time and complexity.

5 Discussion

CAMS-F's evaluation demonstrates its effectiveness in cross-platform CAS development, achieving 35% faster development than traditional methods while supporting over 10,000 IoT devices. The framework successfully balances performance (60 FPS via Flutter [6]) and precision (98% geolocation accuracy [10]), though its DSL requires an initial learning curve [19]. Compared to WebView-based alternatives [16], CAMS-F's native compilation eliminates performance bottlenecks, while its IaC automation [11] reduces deployment errors. Limitations include vendor dependency on Google Maps [10] and the lack of emotional context support — both identified as priority areas for future research. These results advance prior CAS research [1,7] by quantifying the benefits of combining MDD, cloud services, and modern cross-platform tools in real-world applications.

6 Conclusions and Future Work

CAMS-F represents a significant advancement in the development of cross-platform, context-aware applications. By combining Model-Driven Development, cross-platform support with Flutter, and seamless integration with cloud services, CAMS-F simplifies the development process, reduces time-to-market, and improves scalability. Future work includes expanding support for additional IoT capabilities, integrating AI-driven decision-making tools, and further enhancing the DSL to support more complex use cases.

CAMS-F demonstrates that the combination of MDD, Flutter, and cloud services can significantly reduce the complexity of developing cross-platform CAS, with scalability for massive IoT scenarios.

Regarding future work, priority areas: Integrate ROS 2 for advanced robot management in logistics. Extend the DSL to support emotional context (e.g., customer voice analysis). Adopt TensorFlow Lite to predict delivery delays.

References

1. G. Mark, “The Context-Aware Mobile System,” in *Proc. 1st Int. Workshop on Mobile Computing*, 2017.
2. K. Satyanarayanan, “Pervasive computing: Vision and challenges,” *IEEE Pers. Commun.*, vol. 8, no. 4, pp. 10–17, 2001.
3. B. Schilit, N. Adams, and R. Want, “Context-aware computing applications,” in *Proc. IEEE Workshop on Mobile Computing Systems and Applications*, 1994.
4. Gomez-Torres, E., et Al (2025), CAMS: A Model-Based Framework for Context-Aware Mobile Applications with IoT and Azure Maps Integration. Proceedings of ICTIS 2025 New York
5. Gomez-Torres, E., et Al (2025), CAMS-X: Extending the Context-Aware Mobile Systems Framework for Cross-Platform Development with Ionic. Proceedings of ICTIS 2025 Bangkok
6. Google Developers, “Flutter Documentation,” 2023. [Online]. Available: https://flutter.dev/docs
7. Gomez-Torres, E., Challiol, C., & Gordillo, S. E. (2020). Towards a New Perspective of Building Tools for Context-Aware Mobile Applications. In Computational Science and Its Applications – ICCSA 2020 (pp. 567–582). Springer.
8. Firebase. (2023). Firebase Firestore Documentation. https://firebase.google.com/docs/firestore
9. Firebase. (2023). Firebase Cloud Messaging Documentation. https://firebase.google.com/docs/cloud-messaging
10. Google Maps Platform. (2023). Google Maps API Documentation. https://developers.google.com/maps
11. HashiCorp. (2021). Terraform: Infrastructure as Code. https://www.terraform.io
12. Schilit, B., Adams, N., & Want, R. (1994). Context-aware computing applications. Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications.
13. J. Chen and Y. Zhao, Flutter + Firebase para CAS (2023)
14. Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014). Context-aware computing for the Internet of Things. IEEE Communications Surveys & Tutorials.
15. Baldauf, M., Dustdar, S., & Rosenberg, F. (2007). A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing.
16. Ionic Framework. (2022). Build cross-platform mobile apps with web technologies. https://ionicframework.com
17. React Native Documentation. (2023). https://reactnative.dev
18. Flutter. (2023). Beautiful native apps in record time. https://flutter.dev
19. Brambilla, M., Cabot, J., & Wimmer, M. (2017). Model-driven software engineering in practice. Morgan & Claypool.
20. Dey, A. K. (2001). Understanding and using context. Personal and Ubiquitous Computing.
21. Hong, J. I., & Landay, J. A. (2001). An infrastructure approach to context-aware computing. Human-Computer Interaction.
22. Gómez-Torres, E., Challiol, C., Gordillo, S.E. (2020). Variability Features in Building Approaches for Context-Aware Mobile Applications. Information and Communication Technologies of Ecuador (TIC.EC). TICEC 2019. Advances in Intelligent Systems and Computing, vol 1099. Springer, Cham. https://doi.org/10.1007/978-3-030-35740-5_8