



UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Matemáticas y Ciencias Computacionales

TÍTULO: CODE GENERATOR FOR MOBILE APPLICATIONS

Trabajo de integración curricular presentado como requisito para
la obtención del título de Ingeniero en Tecnologías de la
Información

Autor:

Riofrío Valdivieso Andrés

Tutor:

Ph.D. Guachi Guachi Lorena

Urcuquí - Marzo 2020

Urcuquí, 17 de marzo de 2020

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN
ACTA DE DEFENSA No. UITEY-ITE-2020-00011-AD

A los 17 días del mes de marzo de 2020, a las 12:00 horas, de manera virtual mediante videoconferencia, y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dr. CUENCA PAUTA, ERICK EDUARDO , Ph.D.
Miembro No Tutor	Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.
Tutor	Dra. GUACHI GUACHI, LORENA DE LOS ANGELES , Ph.D.

El(la) señor(ita) estudiante **RIOFRIO VALDIVIESO, ANDRES DARIO**, con cédula de identidad No. **1104555303**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **TECNOLOGÍAS DE LA INFORMACIÓN**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-43-No.496-2014**, realiza a través de videoconferencia, la sustentación de su trabajo de titulación denominado: **CODE GENERATOR FOR MOBILE APPLICATIONS**, previa a la obtención del título de **INGENIERO/A EN TECNOLOGÍAS DE LA INFORMACIÓN**.

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

Tutor	Dra. GUACHI GUACHI, LORENA DE LOS ANGELES , Ph.D.
--------------	---

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación a través de videoconferencia, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

Tipo	Docente	Calificación
Miembro Tribunal De Defensa	Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.	7,5
Tutor	Dra. GUACHI GUACHI, LORENA DE LOS ANGELES , Ph.D.	10,0
Presidente Tribunal De Defensa	Dr. CUENCA PAUTA, ERICK EDUARDO , Ph.D.	9,0

Lo que da un promedio de: **8.8 (Ocho punto Ocho)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.


RIOFRIO VALDIVIESO, ANDRES DARIO
Estudiante

Dr. CUENCA PAUTA, ERICK EDUARDO , Ph.D.
Presidente Tribunal de Defensa

Dra. GUACHI GUACHI, LORENA DE LOS ANGELES , Ph.D.
Tutor

Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.
Miembro No Tutor

MEDINA BRITO, DAYSY MARGARITA
Secretario Ad-hoc

Autoría

Yo, **Andrés Dario Riofrío Valdivieso**, con cédula de identidad 1104555303, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor(a) del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Diciembre 2019.



Andrés Dario Riofrío Valdivieso

CI: 1104555303

Autorización de publicación

Yo, **Andrés Dario Riofrío Valdivieso**, con cédula de identidad 1104555303, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Urcuquí, Diciembre 2019.



Andrés Dario Riofrío Valdivieso

CI: 1104555303

Dedication

“To my parents Jimmy Riofrío and Rocio Valdivieso because they give me courage every day of my life. To my brother Jimmy Riofrío because he encourages me to study from high school when he was my best professor. To my sister Andrea Riofrío because despite the distance she was with me every day in these years of study in Yachay Tech.”

Acknowledgements

“To my family that supports keep me away during my studies and to my tutor Lorena Guachi that was able to give me the support, knowledge and patience while I was developing the thesis.”

Resumen

Los generadores de código (CGs) son herramientas de software capaces de crear aplicaciones de forma total o parcial. Su objetivo es mejorar la productividad del desarrollo de software e incrementar la facilidad de entender el código producido. La demanda de los CGs ha crecido drásticamente en los últimos años. En este sentido crear aplicaciones móviles funcionales y de manera rápida ha sido uno de los retos más difíciles en el desarrollo de aplicaciones móviles, ya que el tiempo suele ser mayormente consumido en la creación de las operaciones de manejo de datos: insertar, leer, actualizar y borrar. Por esta razón, en la presente tesis se introduce a la creación de un CG para aplicaciones móviles el cuál se encuentra enfocado en insertar, leer, actualizar y borrar datos. El CG recibe un modelo de datos abstracto como entrada, y provee una interfaz que permite configurar la aplicación móvil. El desarrollo del CG propuesto incluye la selección de un SDK apropiado, arquitectura del software, definición del modelo abstracto y plantillas que establecen la estructura de la aplicación generada. El CG propuesto fue evaluado con métricas de funcionalidad y calidad (rendimiento, accesibilidad, buenas practicas de programación, y optimización de motores de búsqueda). Mientras, que la aplicación móvil fue evaluada en términos de funcionalidad y compatibilidad. Resultados preliminares demuestran que el CG obtuvo 96.67% en calidad y puede ejecutarse en los navegadores web más conocidos como Google, Firefox e Internet Explorer. El CG y la aplicación móvil satisfacen las métricas de funcionalidad. Además, la aplicación móvil satisface los parámetros de compatibilidad por lo que puede ser ejecutada en android mayor o igual a Jelly Bean 4.1.

Palabras clave: Generador de código, aplicaciones móviles, Flutter, XML, Lighthouse.

Abstract

Code generators (CGs) are software tools able to build whole or partial software applications. They aim to improve software development productivity and increase code understandability. The demand for mobile applications has grown dramatically in the last few years. In this sense, it is essential to create successful mobile applications every time, faster, and face one of the most challenging issues. These issues are in the field of mobile application development, which is the high time committed to writing the code to insert, read, update, and delete data. For this reason, the present thesis introduces a CG for mobile applications focused mainly on producing code to insert, read, update, and delete data. It receives an abstract data model as input and provides a data settings user interface to build a mobile application. The development of the proposed CG includes the selection of the appropriate SDK, software architecture, definition of the abstract model, and templates to establish the structure of the generated mobile application. Functionality and quality metrics (performance, accessibility, best programming practices, and search engine optimization metrics) were used to evaluate the efficiency of the proposed CG. Preliminary results demonstrate that CG achieves 96.67% in quality metrics and can run on the most well-known web browsers such as Google, Firefox, and Internet Explorer. Furthermore, the CG and generated mobile application satisfy the functionality metrics. Besides, the mobile application satisfies the compatibility metrics. Then, it can run in android devices with android greater or equal than Jelly Bean 4.1.

Keywords: Code Generator, mobile applications, Flutter, XML, Lighthouse.

Contents

Acronyms	9
1 Introduction	11
1.1 Problem Statement	12
1.2 Justification	13
1.3 Contribution	13
1.4 Thesis Overview	14
2 Objectives	15
2.1 General Objective	15
2.2 Specific Objectives	15
3 Literature Review of CG for Mobile Applications	16
3.1 Code Generator (CG)	16
3.1.1 Background and History of CGs	17
3.1.2 Modeling of a CG	17
3.1.3 Code Generator Approaches	18
3.1.4 CGs Types	20
3.1.5 Advantages and Disadvantages	22
3.2 Mobile Applications	23
3.2.1 Background and History of Applications	23
3.2.2 Software Development Kit (SDK) for Mobile Applications	24
3.2.3 CG For Mobiles	30

4	Methodology to Develop a CG for Mobile Applications	32
4.1	Planning	34
4.1.1	Definition of CG Functionality	34
4.1.2	Definition of Mobile Application Functionality	35
4.1.3	SDK Selection	36
4.1.4	Architectural Pattern Selection	37
4.1.5	Templates definition for Mobile Application Interface and Code	39
4.1.6	Abstract data model: File format	44
4.1.7	Implementation Constrains given by Selected Tools	45
4.1.8	Constrains to Implement the CG and Mobile applications	45
4.2	Engineering	46
4.2.1	Design	46
4.2.2	Implementation	48
4.3	Evaluation	49
5	Experimental Setup	50
5.1	CG Metrics	50
5.1.1	CG Functionality	50
5.1.2	Performance Metrics of CG Web Page	51
5.1.3	Accessibility Metrics of CG Web Page	51
5.1.4	Best Practices Programming Metrics of CG Web Page	52
5.1.5	SEO Metrics of CG Web Page	54
5.2	Application Metrics	55
5.2.1	Application Functionality	55
5.2.2	Application Compatibility	55
5.3	CG Experiments	56
5.3.1	Experiment 1: Functionality of the CG.	57
5.3.2	Experiment 2: Analysis of performance, accessibility, best practices programming, and SEO metrics of the CG through Lighthouse tool. . .	57
5.4	Mobil Application Experiments	58

5.4.1	Experiment 1: Functionality of the mobile application.	58
5.4.2	Experiment 2: Compatibility of the mobile application.	58
6	Results	59
6.1	CG Results	60
6.1.1	Experiment 1: Functionality of the CG.	60
6.1.2	Analysis of performance, accessibility, best practices programming, and SEO metrics of the CG through Lighthouse tool.	60
6.2	Application Results	63
6.2.1	Functionality of the mobile application	63
6.2.2	Compatibility of the mobile application.	64
7	Conclusions and Future Work	65
	References	68
	Appendices	76
A	UML Diagram of the Flutter Recipe Cost Developed by the CG Created.	78
B	User Manual	80
B.1	Update XML Postgres Database.	81
B.2	Complete Application Information.	81
B.3	Run the Application	85
C	Technical Manual	87
C.1	Installation of the CG	88
C.2	CG User Requirements	89
C.3	Templates definition for Mobile Application Interface and Code	92
C.4	Architectural Pattern	96
D	Thesis Web Site	101

List of Algorithms

4.1	PHP template to connect database with apache server.	42
4.2	Java code to replace PHP template of the Algorithm 4.1.	42
4.3	Flutter template to create buttons of the main window of the mobile application.	43
4.4	Java code to replace Flutter template of the Algorithm 4.3.	44
C.1	PHP template to connect database with apache server.	94
C.2	Java code to replace PHP template of the Algorithm C.1.	94
C.3	Flutter template to create buttons of the main window of the mobile application.	95
C.4	Java code to replace Flutter template of the Algorithm C.3.	96

List of Figures

3.1	Top-down process of MDSE methodology [1].	18
3.2	Structure of CG inside a compiler [2].	19
3.3	Example of structure of CG for visual objects, UML diagrams, SDL to obtain Java and C++ code [3].	19
4.1	Methodology used to create a mobile application from CG	33
4.2	Spiral model for a CG.	33
4.3	CG front-end used to configure mobile application settings.	35
4.4	MVC architecture for the CG	38
4.5	MVC for mobile applications developed with the CG	38
4.6	Design of a Standard UI based on definition DEF_1-DEF_3. a) tablePageSet- ting.jsp and b) Mobile application.	40
4.7	Standard update and insert windows for mobile applications based on DEF_4 . .	41
4.8	Standard manage of primary and foreign keys based DEF_5 and DEF_6. Fur- thermore, standard confirmation message before to make an insertion in the database from mobile application based on DEF_7.	41
4.9	Package diagram of the CG	47
4.10	Package diagram of the mobile applications.	47
4.11	Deployment diagram of CG.	48
4.12	Deployment diagram of mobile application.	48
6.1	Sequence diagram to create an application with the CG developed.	59

A.1	UML diagram of the Flutter recipe cost application obtained by the CG.	79
C.1	Sequence diagram to create an application with the CG developed.	88
C.2	CG user requirement number UCGR_1	89
C.3	CG user requirement number UCGR_2	89
C.4	CG user requirement number UCGR_3	90
C.5	CG user requirement number UCGR_4	90
C.6	CG user requirement number UCGR_5	91
C.7	CG user requirement number UCGR_6	91
C.8	CG user requirement number UCGR_7	91
C.9	Design of a Standard UI based on definition DEF_1-DEF_3. a) tablePageSetting.jsp and b) mobile application.	92
C.10	Definition DEF_4.	93
C.11	Definition DEF_5 until DEF_7.	93
C.12	MVC architecture of the CG.	97
C.13	Package diagram of the CG.	97
C.14	Class diagram of the CG.	98
C.15	MVC for mobile applications developed with the CG	99
C.16	Package diagram of the mobile applications	99
C.17	Class diagram of the mobile applications	100

List of Tables

3.1	Examples of CGs developed.	22
3.2	Resume of comparison between SDKs to generate mobile applications.	29
3.3	Comparison among CGs for mobiles.	31
4.1	Spiral Model planning summary to create a CG for mobile applications.	34
4.2	Architectural pattern features for CG and mobile applications.	37
4.3	Tools definition for CG and mobile applications.	45
5.1	Metrics for functionality of the CG.	50
5.2	Weights for “tableSetting.jsp” performance.	51
5.3	Weights for “tableSetting.jsp” accessibility.	52
5.4	Weights for “tableSetting.jsp” best programming practices.	53
5.5	Weights for “tableSetting.jsp” SEO.	54
5.6	Metrics for functionality of the mobile application.	55
5.7	Metrics for compatibility of the mobile application.	56
5.8	Server and Client used for CG experiments	57
6.1	Results of CG, experiment 1: Functionality.	60
6.2	Results of CG, experiment 2: Performance.	61
6.3	Results of CG, experiment 2: Accessibility.	61
6.4	Results of CG, experiment 2: Best practices programming.	62
6.5	Results of CG, experiment 2: SEO.	63
6.6	Results of mobile application, experiment 1: Functionality.	63

6.7 Results of mobile application, experiment 2: Compatibility. 64

C.1 CG Paths to update. 88

Acronyms

API Application programming interface. 53, 55, 62

CG Code Generator. 1–3, 5–8, 11–23, 25, 28, 30–39, 42, 44–52, 54–57, 59–63, 65, 66, 78, 79, 88–92, 94, 96–99

CPU Central processing unit. 46, 51, 61

CSS Cascading style sheet. 24

GPS Global positioning system. 25

HTML Hypertext markup language. 24, 26–29, 37, 53, 62

Http HyperText Transfer Protocol. 53, 54, 62, 63

IDE Integrated development environment. 46, 88

ISML International Saimoe League. 22

Java EE Java Enterprise edition. 42, 44–46, 59, 88, 94, 96

JDK Java development kit. 46

MDA Model Driven Architecture. 17

MDSD Model-driven software development. 17

MDSE Model-driven systems engineering. 5, 17, 18

MVC Model-view-controller. 5, 6, 30, 37, 38, 45, 59, 65, 96, 97, 99

OMG Object Management Group. 17

OS Operative system. 23, 24

OSS Open source. 20–22

PDA Personal digital assistance. 23

PHP Personal Home Page. 21, 22

SDK Software Development Kit. 1, 7, 15, 23–30, 32, 34, 36, 37, 65

SDL Simple DirectMedia Layer. 5, 19

SEO search engine optimization. 2, 7, 15, 54, 57, 60–63, 65, 66

UI User interface. 24, 27–29, 31

UML Unified modeling language. 3, 5, 6, 17, 19, 78, 79

XML Extensible Markup Language. 3, 59, 81, 88, 96

Chapter 1

Introduction

Software companies aim to develop large applications in a short time. For this reason, CGs software applications that produce code faster have been incremented from 1989. These CGs generate code from templates (static code used as structure of different applications [4]) to create: mobile application, desktop application, PHP code, etc. The most CGs are focused on writing code automatically for web platforms, and only a few generators have been developed for mobile devices. Some of the existing CGs produce code for different types of applications such as Genexus CG [5] and Code On Time [6]. Genexus allows us to generate web and mobile applications, and Code on Time produces mobile and desktop applications. Meanwhile, others generate specific applications as: PHPMaker [7] produces PHP code, CodeCharge [8] and Scriptcase [9] produce web applications, and SoftFluent CodeModeler [10] to create .Net applications.

In the last years, the increment of smart devices increases the need for development faster and successful mobile applications. For this reason, one of the most significant expectations of CGs is to develop entire applications able to be modified to solve bugs or add new functionalities. Nevertheless, almost all CGs return only the executable application and not the source code. Particularly, in the field of software development, one of the most difficult issues is the high time committed to writing the code to insert, read, update, and delete data.

In this sense, the current work presents a methodology to develop a CG for mobile applica-

tions focused mainly on producing code to insert, read, update, and delete data. It receives as input an abstract data model and provides a user interface for data settings. The development of the proposed CG includes the selection of the appropriate SDK, software architecture, definition of the abstract model, and templates to establish the structure of the generated mobile application. In addition, the proposed CG is free and open-source and returns the source code to allow developers to fix issues from the CG or mobile application code.

The current work tests the quality CG in terms of functionality and quality metrics (performance, accessibility, best programming practices, and search engine optimization metrics). The functionality evaluation is supplemented by the generated mobile application tests to determine the ability of the produced mobile application to insert, read, update, remove, and search data effectively using the created mobile user interfaces.

Preliminary results show that the proposed CG approach is functional and potentially can save valuable time in the development of transactional applications where common repetitive codes have gained an important role to insert, read, update, and delete information. The CG and generated mobile application satisfy the functionality metrics successfully. The CG achieved a grade of 96.67% in quality metrics and the mobile application satisfies android compatibility metrics. In this way, the CG and mobile applications do not need functionality changes, and they are ready for use. The CG works in some browsers like Google Chrome, Firefox, and Internet Explorer, and the mobile application functions in android greater or equal than Jelly Bean 4.1.

1.1 Problem Statement

CGs are becoming indispensable to produce code in less time and cost. In particular, the insert, read, update, and delete operations could be used inside templates of the CG to avoid writing these transactional operations. A few CGs are focused on developing transactional mobile applications. However, not all they return the source code to the developer could make changes.

Although CGs have been used to reduce the time and cost of development of transactional mobile applications such that tourism, commerce, banking, among others. The development of CG is still of great interest in improving productivity by reducing effort in the repetitive code,

promoting the later reusability, readability, and maintenance. This topic is challenging, and there is no complete, automatic, and adequate CG.

1.2 Justification

CGs are desirable to create entire applications like mobile applications, desktop applications, web applications, smartwatch applications, among others. To develop specific code like PHP code, Flutter code, SQL code, etc., where the transactional applications have gained an important role to insert, read, update, and delete information used in entire applications or specific code.

In recent years, mobile applications that manage data (insert, read, update, and delete) increased to 80% [1]. Therefore, this thesis aims to develop a free and open-source CG to automatically produce a mobile application that works with transactional operations like insert, read, update, and delete information from a database. This work is not restricted to understand, adapt, and create CGs for transactional operations (insert, read, update, and remove data); it also carries on new research challenges for those who are focused on this field. In this sense, other programmers could augment the proposed CG or customize it to generate applications for one specific type of business. It also provides the opportunity to do more in-depth explorations for software development or even commercial fields in automatic code generation.

1.3 Contribution

This work aims to develop a CG that receives an abstract data model to automatically produce the code of a mobile application with its corresponding mobile user interfaces to insert, read, update, delete, and search data. For this, an appropriate methodology has been established. As well as proper mobile SDK and architecture pattern has been selected focused on build a functional and multi-platform mobile application. Besides, to do experiments in a real scenario, an application to obtain the recipe cost of the restaurant industry was created by using the proposed CG.

From preliminary results, information related to functionality and quality (performance,

accessibility, best programming practices, and search engine optimization metrics) is provided to both researchers and software development engineers to determine whether the proposed approach meets the functionality expectations.

1.4 Thesis Overview

This work is divided into seven main Chapters named as follows: 1) Introduction, 2) Objectives, 3) Theoretical Framework of CG for Mobile Applications, 4) Methodology, 5) Experimental setup, 6) Result, and 7) Conclusions and Future Work.

Chapter 1, it describes the problem statement, the justification and the developers' contributions.

Chapter 2, it presents the general and specific objectives.

Chapter 3, it states an overview of CGs and mobile applications.

Chapter 4, it is the methodology and gives CG and mobile application functionality. Furthermore, iterations of the spiral model used to develop the CG is described.

Chapter 5, it defines the metrics to analyze the CG and mobile application. Furthermore, it defines the experiments to evaluate them.

Chapter 6, it displays the results obtained when the experiments of the previous chapter are used.

Chapter 7, it presents the conclusions and future work that could be developed when the methodology is improved and gives some possible issues.

Chapter 2

Objectives

2.1 General Objective

To develop a code generator to produce mobile applications for android and iOS able to insert, read, update, and delete information from a database system using an SDK that allows to produce code with high accessibility, usability and performance.

2.2 Specific Objectives

- To establish the architecture pattern for CG and generated mobile application.
- To select software and hardware tools to develop the CG.
- To determine an appropriate SDK to be used by the CG to produce mobile applications.
- To evaluate the quality of the CG in terms of functionality, performance, accessibility, good programming practices, and SEO scores.
- To evaluate the quality of the mobile application in terms of functionality and compatibility.

Chapter 3

Literature Review of CG for Mobile Applications

A Code generator (CG) is a program that generates programming code without the need of a programmer. Its primary purpose is to shorten the time required by the software development phase. This chapter presents a brief description of existing CG starting with its background and history. Then, several approaches with their advantages and disadvantages are explored. Finally, mobile applications, its background and the CGs created for them are stated.

3.1 Code Generator (CG)

A CG or a model compiler is a software that generates code using templates and an abstract data model, to produce software code of another application (web, desktop, mobile). This task could be compared with compilers that can transform high-level code into low-level code [1]. A template is a static and repetitive text used inside the different applications, which serves as a code structure in a dynamic way [4]. Meanwhile, the abstraction for define how a system shall be implemented is known as “model” in the CGs context [1]. The current section describes the background and history, the modeling stage, the approaches, the types, the advantages and disadvantages of CGs.

3.1.1 Background and History of CGs

The first CGs were the compilers. They are able to generate machine language using instructions as model [11], such as GeneXus that born in 1989, and it is a cross-platform tool with more than 30 years in the market. Furthermore, Genexus allows creating cross platforms for web and mobile applications and their databases from its language [12].

In 1989, the Object Management Group (OMG) was founded, and in 2001, it launched the Model Driven Architecture (MDA) as a tool for translating UML model elements into code without taking account development platforms [13]. However, at the same time, new CG alternatives appeared to use databases as a model [14, 8, 9].

After 2001 the CGs started to use its own languages, databases or visual objects as models in order to create software (see Table 3.1). Among the software created by CGs are: mobile applications [15, 16, 17], plugins [18], desktop application [6], and web [19, 20, 21].

Currently, the CGs are used to created software from a model. However, the model concept has been changed. Now, the model is an abstract technical solution to resolve systems engineering problems [22] that are essential in the process of automatic code generation because models allow creating code trough their objects [4].

3.1.2 Modeling of a CG

Modeling is a process of abstract the reality in a model allowing focus on interest aspects (reduction features). It generalizes the characteristics of a prototype (mapping features) to define how shall be implemented an application [1]. It is often applied before implementing a system (produce code manually or automatically using a CG). Currently, model-driven systems engineering (MDSE) and model-driven software development (MDSD) are the most common methods for modeling a system in the software developing because both are used from planning to the implementation process. However, MDSD does not have an essential role because its models do not represent the functionalities of the program for the developing process as MDSE that are most used in CGs [23].

The MDSE methodology consists in conceptualize and implement a system, see Fig. 3.1. The conceptualization can be performed through three forms: 1) Application, where models

are abstracted and represented by a set of diagrams, for instance, UML diagrams, rules of transformation, and components. 2) Application Domain that defines the modeling language, rules of transformation, and platform used. Finally, 3) Meta Level defines the language and its operation in a low-level form. Meanwhile, the implementation can be done by the following forms: 1) Modeling to define the models. 2) Automation to read models and write them in language that could be read by the system, and 3) Realization to implement a solution from artifacts as functions and methods [1].

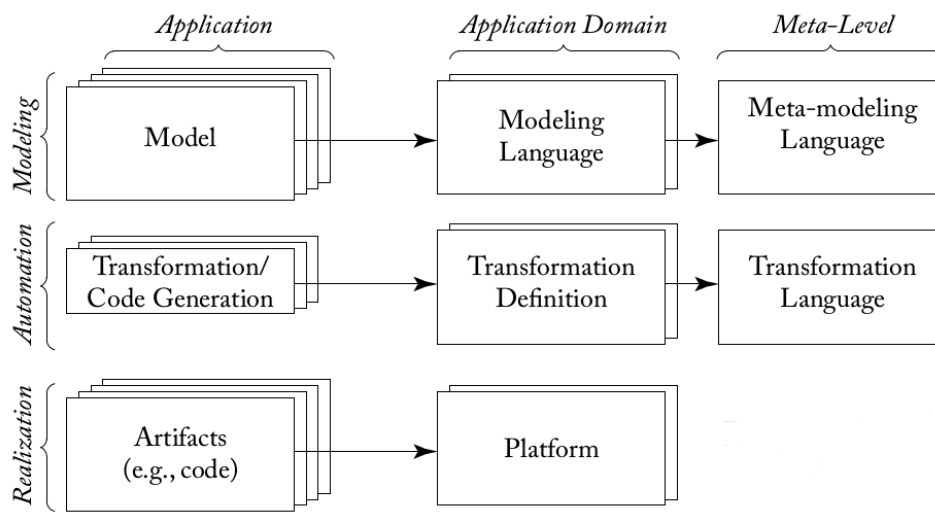


Figure 3.1: Top-down process of MDSE methodology [1].

3.1.3 Code Generator Approaches

The term CG appeared with the compilers to translate the human language to machine language [24]. The compiler idea was born with Rutishauser, in 1952 [24] by using an intermediate language to perform machine operations to produce automatically a program written in machine code [11, 2] as shown in Fig. 3.2 that start with intermediate code to produce executable code. CG outline is still used in some software development companies, to use an intermediate language to translates high-level into low-level language and not to machine language like compilers. Then, the software produced uses more high-level languages avoiding the use of low-level languages that needs more time to build a similar application.

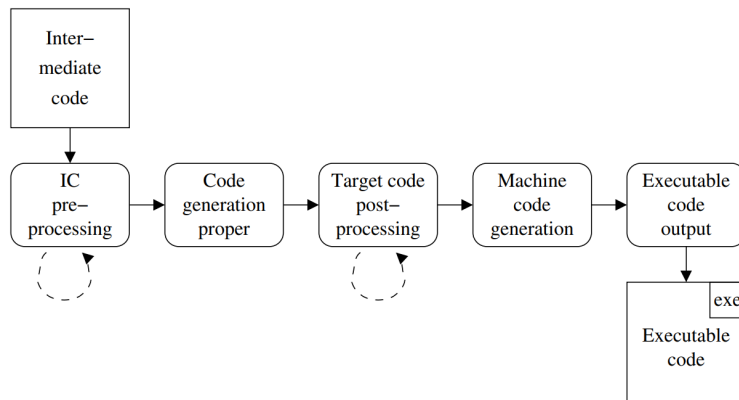


Figure 3.2: Structure of CG inside a compiler [2].

In order to generate software code automatically, a CG often uses models based on visual object [25], UML diagrams [26], or databases [14] as a conceptual definition that contains code that will be generated for each object, table, etc. Those models are translated to metadata, and with some rules operations (as every string will have a label or textbox), the final code is generated, or the software is obtained (see Fig. 3.3) [3]. CGs that works with a database can connect to them to make a Syntactic Analysis or receive the metadata to start from Semantic Analysis of the Fig. 3.3.

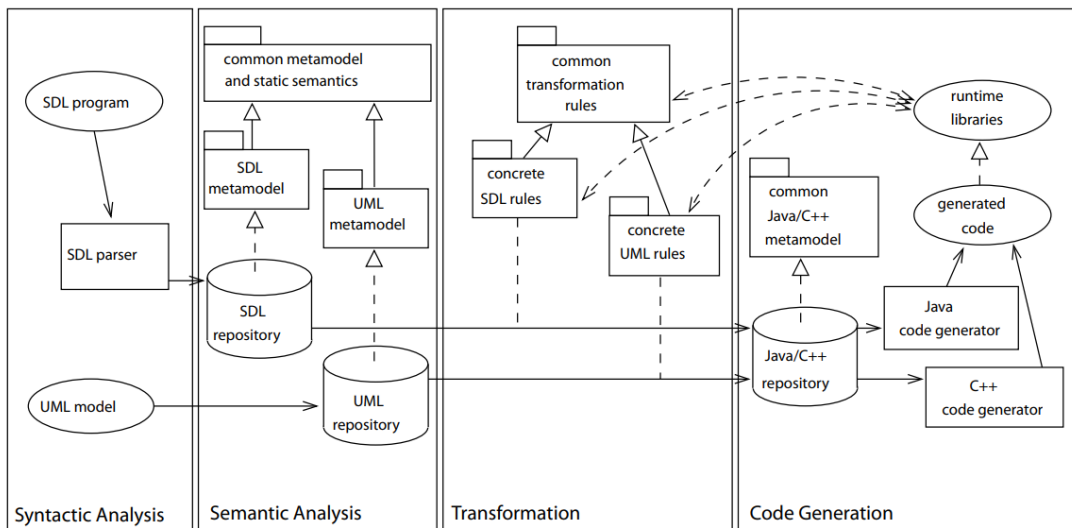


Figure 3.3: Example of structure of CG for visual objects, UML diagrams, SDL to obtain Java and C++ code [3].

CG approaches mainly have been focused on produce full code of applications for organizing objects, products, and people such as make inventories, manage employees, obtain weather information, monitoring prices, manage health records, and so on [27]. However, there are round-trip engineering CGs that produce code for specific requirements [4] as the CGs that work with visual objects that create the front-end of applications like activities, colours, buttons or text style that could be used in templates (see Table 3.1). On the other hand, some approaches are strictly focused on insert, read, update, and delete operations due to 80% of application that manages data use them [1].

3.1.4 CGs Types

Different types of CG have been introduced in the literature, each of them for specific aims such as to be open source, work on a specific platform or device, platforms of code generated, and the model or object used. Those distinctive characteristics of different CG types are furnished in Table. 3.1. This table allow to determine if a CG is open source (OSS), free, how it works, the code that produces, the model or object used, and when it was launched. Base on the summary depicted in Table. 3.1, it can be seen that the most of CGs work from an object that works in a visual form (users assemble each activity from the object) or CGs work directly with a modeling language that allow to create each activity faster than visual objects. Then, Table 3.1 shows that CGs are used to develop web and mobile applications with CGs that were created from 1989 until 2017, and they are working until now. Furthermore, there is a small portion of CGs that are free and open source.

CG name	OSS	Free	Work as	Code produced	pro-	Model/Obj	Launched
GeneXus [5]	No	No	Desktop (Windows)	Mobile, Web		Genexus language	1989
PHPRunner [14]	No	No	Desktop (Windows)	Web		Database	1999
CodeCharge [8]	No	No	Desktop (Windows)	Web		Database	2000
Scriptcase [9]	No	No	Web, Desktop (Mac, Windows)	Web		Database	2000
AppGini [28]	No	No	Desktop (Windows)	Mobile, Web		Database	2002
Iron Speed Designer [19]	No	No	Desktop (Windows)	Mobile, Web		Database	2003
PHPMaker [7]	No	No	Desktop (Windows)	PHP		Database	2002
dbQwikSite [20]	No	No	Desktop (Linux, Windows)	Web		Database	2004
SoftFluent Code-Modeler [10]	No	No	Desktop (Windows)	.NET		Database	2005
Code On Time [6]	No	No	Desktop (Windows)	Mobile, Desktop		Database	2008
SynApp2 [29]	Yes	Yes	Web, Desktop (Linux, Mac, Windows)	Web		Database	2008
MOBILEROADIE [30]	2009	No	No	Web		Visual Objects	2009
Zathuracode [21]	Yes	Yes	Plugin	Web		Database	2009
AppMakr [31]	No	No	Web	Mobile		Visual Objects	2010
Shouteín [32]	No	No	Web	Mobile		Visual Objects	2010
AppMachine [33]	No	No	Web	Mobile		Visual Object, Web	2011
GoodBarber [34]	No	No	Web	Mobile		Visual Objects	2011
TheAppBuilder [35]	No	No	Web	Mobile, Web		Visual Objects	2011

CG name	OSS	Free	Work as	Code produced	pro-	Model/Obj	Launched
appery.io [36]	No	No	Web	Mobile, Web		Visual Ob-jects	2013
Appy Pie [37]	No	No	Web	Mobile		Visual Ob-jects	2013
Backendless [38]	No	No	Web	Mobile, Web		Visual Ob-jects	2013
CodeBhagat [39]	No	No	Desktop (Windows)	.NET		Database	2014
App Builder [40]	No	No	Web	Mobile		Visual Ob-jects	2015
Anchurus-GEN [41]	Yes	Yes	Plugin	PHP		ISML	2016
Flutter Studio [25]	No	Yes	Web	Mobile		Flutter Wid-gets	2017

Table 3.1: Examples of CGs developed.

3.1.5 Advantages and Disadvantages

The benefits of using a CG could be achieved after the developers learn how to use this tool and manipulate its output. However, it is essential to show the benefits and obstacles that exist when a CG is used in a developing company.

Advantages

- The CG allows us to generate code for different languages (Java, C++, Python) or machines (smartphones, tablets, laptops) [4], thus saving the developer from having to use of a specific language to create an application for a particular device.
- CG works with verified models, the reliability and robustness of this is improved avoiding fix error in the developing process [4] [1].
- A CG allows to develop a prototype automatically with different software architectural patterns, and according to developer necessities [1].
- Modules and new characteristics could be implemented in CGs [4] [42] to give new characteristics at programs created by CG.

- The final code obtained is another advantage of CG because it has created according to the style guide of the programming language [1] [23].

Disadvantages

- The CG standard style of writing could be unfamiliar [23].
- Make changes could be three times more difficult in CG code than in your code [43].
- CG requires adaptation to use and this could involve costs of training [44] [45].
- It is necessary to use more time to construct the model [46] for the CG.
- The curve of CG learning could be difficult to overcome [42].

3.2 Mobile Applications

A mobile application often also called Mobile application is a software application developed to run on any mobile and personal digital assistance (PDA) [47]. At the beginning, mobile applications were developed for personal and productivity assistance as calendar, e-mail, audio, email, internet and social web site [48], but they were expanded rapidly due to the public and wide demand in several areas as e-commerce, ticket purchases, order-tracking, location services, health control [49]. That is the reason why currently there are more than 2.46 million of applications in the Google Play Store and 1.96 million of applications in the Apple App Store to download [50]. In this section the history of applications, SDKs to construct applications, and CGs used for creating mobile applications are described.

3.2.1 Background and History of Applications

The Standalone Java services as games or calendars born in 2002 [51]. In 2005 distributed Java services appeared like video player from internet and office applications were launched [51]. In early 2007 iPhone introduces the first smartphone to the market and in the late of this year Google makes public its Android OS; in 2007, the services of smartphones increased, they

have features like audio, email, internet and social web site [48]. However, the timeline of the developed applications is a black box after 2005 [51].

With the introduction of smartphones, new operative systems born as iOS and Android that comes with its programs denominated like applications (apps); this OSs allow installing new application developed by the manufacturer of the smartphones or third parties. There are different categories of mobile applications according with its characteristics as books, business, comics, communication, education, entertainment, finance, health, libraries, lifestyle, video player, medical, musical/audio, news/magazines, photography, productivity, shopping, social, sport, transport, travel, weather and so on [49].

3.2.2 Software Development Kit (SDK) for Mobile Applications

SDKs is a software to write custom applications through tools as libraries and compilers [52]. This applications could be written in SDKs to native application or SDKs to cross platforms (different operative systems) [53]. The mobile SDKs could be classified as native, web views [54], native user interfaces [55] and 2D rendering engine [56].

- **Native SDK:** It compiles in a language known by the operative system [57]. The best-known SDKs are Android and iOS [58].
- **Web views SDK:** It creates mobile web applications with broadband mobile access, buttons and all the other characteristics that have web pages programmed with CSS, HTML and JavaScript standards as Apache Cordova and Adobe Systems [59]. Unfortunately, this type of application runs in a web browser and not use the specifics components of the platform. However, it is possible to implement web view components inside a native application as in Ionic SDK, but the performance is reduced [60].
- **Native user interface SDK:** It can develop applications using packages that could interpret native components of the cell phones using specific languages as C# or Java Script [61]. Between SDKs available to produce native UI we have Xamarin property of Windows and React Native created by Facebook [61].

- **2D rendering engine SDK:** It transforms the code to C++ and can be rendered with Skia [56] that is a 2D graphics library written in C that is used in Chrome, Firefox, Android, iOS and other products [62].

The last three types of SDKs are able to produce cross-platform applications that could be the bases of a CG to obtain quickly and structured code. Evaluate the advantages of SDKs are significant from the developer point of view because it facilitates the creation of the application [63]. Therefore, it is essential to analyze SDKs variables like access to system features, graphical performance, level of modularization, programming language, and use interface that could be obtained. Access to system features refers to the accessibility of different components as GPS or camera; the graphical performance tries to compare the speed of mobile graphics created in the SDKs, programming language try to analyze the most popular programming languages to developers, user interface refers at the behavior and how looks the applications [54]. Furthermore, the variable level of modularization of the programming language is analyzed because it lets reuse code in the CG from high-level models [64]. This comparisons are summarized in the Table 3.2

Access to System Features

Mobile functionalities as camera or GPS can be accessed directly by native SDKs, but not by web views that are inside of native application as Cordova that needs plugins written in JavaScript, while web views that are outside of native application not can have access to system features [63]. In the case of the native user interface has not directly access to camera or GPS but there are third party plugins that let make it connection [65]. In the case of Flutter that is a 2D rendering engine has their plugins in dart languages supported by Google to do its connections [66] and it accept third party plugins for some features that not have access [67].

Graphical Performance

Native SDKs has high performance; while, inside web views have relatively high performance and outside web views have low-performance [68]. Native user interfaces represent by React

Native has high performance and 2D rendering engine has high and more stable performance than React Native [69].

Level of Modularization

The modularization is more difficult in languages that are not oriented to objects as HTML by this reason all the web views have less level of modularity than native, native User Interfaces and 2D rendering engine. Furthermore, Flutter has more modularity than the rest of SDKs because all objects are widgets [56] and works with dart that has all that you can place in variables as an object for example numbers or null object [70].

Programming Language

To know what languages are more known in the world by developers, in this thesis we say that the top 20 in the TIOBE list are the most known, the next 30 are moderately known, and the other 205 are little known [71]. Furthermore, as HTML is not considered as a language in TIOBE, this thesis collocates it in the top because almost all the papers reading about web views say that developers know about this type of programming. Then, the language of native SDKs is most known same that web views and native user interfaces and 2D rendering engine is moderately known.

User Interface

Currently has an interface that looks good and is friendly with the user is essential. Native SDKs let create functional interfaces, and the native web views could be homologous than native but with web components [63] while native user interfaces could use all components of the cellphone to create a good interface [72] and 2D rendering engine allows creating beautiful interfaces because use Material Design and Cupertino widgets to develop it [56].

Development Cost

The companies of developer need reduce cost and maintain high qualities levels of products to be competitive [73] by this reason is essential to analyze attributes that affect the price of devel-

oping as compatibility, distribution, and upgrade, licenses and time of reloading the application in cellphones. Compatibility refers to supported mobile platforms, distribution and upgrade is the form that user could obtain the application and obtain updates of versions, licenses refer to permissions with or without the cost of uses of SDKs to create commercial or free software [54]. Furthermore, the time of reloading the application in cellphones refers to the time of installation the applications in the mobile by the developer to see changes while he is programming [63].

Compatibility

When programming in Android Studio or Xcode, we have programs for one operative system according to the SDK that we chose; then in native SDKs are fragmentation problems (different operative systems versions that need different lines of codes to program the same thing) [63]. In web views are possible to develop for Android, iOS and Windows phone if we use Cordova [74] or Phonegap [75]. Android, iOS, and Blackberry if we use Titanium, or all operative system that supports HTML5 if we use outside native web views [76]. If you use native UI, you could program to Android and iOS [65] same that in 2D rendering engine SDK [56].

Distribution and Upgrade

The outside web views distribution and upgrading is through at URL because this types of applications are web pages adapted to cellphones that we could open in almost any web navigator; however, to install and upgrade new versions of programs in native, inside web views, native UI, and 2D rendering engine SDKs is through application stores [54] as the play store that has a cost of \$25 [77] or apple store that has a price of \$99 per year [78].

Licenses

When we like a program in native SDK, we could download a free program to Android as Android Studio [79] or paid program to iOS as Xcode [78]. To web views is possible find open source and free programs [54] same to native user interfaces [65] and 2D renderig engine [80].

Time of Reloading the Application in Cellphones.

The possibility of doing editions and see changes in the cellphone instantaneously (hot reloading) reduce the time of developing the applications, this possibility we could find in native UI [81] and 2D rendering engine [56]. For other SDKs the time of observing changes in the cellphone is same to install the application: in Native applications is about 7.64 seconds and to web views SDKs 2 or 3 times bigger than native, however with Cordova SDK is 9.37 seconds [63].

Security

With HTML5 the security problems of webs and cellphones increase and with the use of new JavaScript versions is even more insecure; furthermore, the use of third party plugins has been reducing because of increasing the forms that users could be attacked [82]. Then, as the CG is a focus on the creation of transactional applications, this subsection gives you a perspective to possible security problems that could have applications created by the CG proposed.

Use of HTML5.

Web views use as language HTML5 that could cause security problems even more if they use JavaScript [54]. Then, to avoid this problem is possible to develop mobile applications using native SDK, native UI SDK or 2D rendering engine SDK.

Use of Third Party Plugins.

The use of third party plugins depends on the applications; it has more necessity of plugins when the application needs to has access to components of the system as in subsection of access to system features. Then according to the passage reading before, we could say that native SDKs does not need third party plugins, 2D rendering engine needs sometimes, and the others SDKs need almost always use third-party plugins when try to use components of the system.

Variables	Native	Web Views		Native UI	2D rendering engine
		Inside Native	Outside Native		
Advantages of SDK					
Access to system features	direct	plugins	unavailable	plugins	direct and third party
Graphical performance	high	relatively high	low	high	high and more stable
Level of modularization	high	low	low	high	very high
Programming language	most know	most know	most know	most know	moderately know
User interface	good	good	good	good	good
Cost of develop					
Compatibility	Android or iOS	Android, iOS, Other	Android, iOS, Other	Android, iOS	Android, iOS
Distribution and upgrade	application stores	application stores	URL	application stores	application stores
Licenses	free, paid	free, open source	free, open source	free, open source	free, open source
Reload application	7.64s	9.37s or more	2-3 times more than native	instantaneous	instantaneous
Security					
Use of HTML5	no	yes	yes	no	no
Third-party plugins	never	almost always	almost always	almost always	sometimes

Table 3.2: Resume of comparison between SDKs to generate mobile applications.

3.2.3 CG For Mobiles

A CG for mobiles takes account the SDK that will be used to develop mobile applications. Furthermore, the MVC architectural pattern is used to create the most CGs and return the code with this pattern too [21]. Then, to know which are the best CGs for mobile applications is necessary to analyze the SDK that use them, using Table 3.2. Moreover, To determine if the mobile CGs of Table 3.1 could be full CGs or round-trip engineering CGs is necessary to analyze if they produce insert, read, update, and delete operations (see Section 3.1.3). Besides, the maintenance of the application is essential because it reduces costs [83] and the code to edit the CG application is vital to developers when there are problems as bugs as shown in Table 3.3.

Therefore, the Backendless CG is the best of Table 3.3 because it gives the opportunity of works with all types of SDKs, allowing approach their best qualities according to application necessities. Furthermore, it produces full applications and maintains them. Moreover, this CG gives the code to developers, and it could manipulate this when it is necessary (for example to resolve problems or edit application logic that could no be created by the CG).

CG name	SDK			Read, insert, update and delete data	Maintenance	Return Code	Return App
	Native	Web View	Native UI				
GeneXus [5]	✓	✓		✓	✓	Web	✓
AppGini [28]		✓		✓	✓	App	
Iron Speed Designer [19]		✓		✓	✓	Web	
Code On Time [6]		✓		✓	✓	Web	
AppMakr [31]	✓	✓			✓		✓
Shouterh [32]			✓			App	
AppMachine [33]	✓				✓		✓
GoodBarber [34]	✓	✓			✓		✓
TheAppBuilder [35]			✓		✓	App	
appery.io [36]		✓		✓	✓		✓
Appy Pie [37]	✓				✓		✓
Backendless [38]	✓	✓	✓	✓	✓	App, Web	
App Builder [40]	✓	✓					✓
Flutter Studio [25]				✓		App	

Table 3.3: Comparison among CGs for mobiles.

Chapter 4

Methodology to Develop a CG for Mobile Applications

Mobile applications have drastically influenced the way many businesses work. In recent years, the restaurant world has demanded significantly mobile applications development. Those applications have helped to restaurant owners to provide faster attention, solidify their presence, and consolidate their process. In this sense, the current chapter presents the methodology that consists of three stages planning, engineering and evaluation to create a CG to produce mobile applications code as is shown in Fig. 4.1. For validation purposes, the proposed CG will be used to read, insert, update, and delete information used in the process of determining the restaurants' recipe costs.

The planning stage of the methodology consists into determining functionalities of CG and mobile application, SDK, tools, constraints, and templates. The engineering stage includes two substages transformation (where the model is read) and code generation (mobile application is created). The evaluation stage determines if the CG and mobile application work according to metrics defined in Chapter 5.

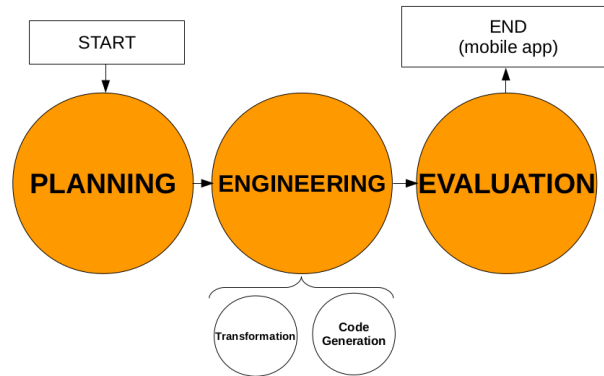


Figure 4.1: Methodology used to create a mobile application from CG

To develop the methodology defined in Fig. 4.1, the spiral model was chosen to generate the CG from Software Engineering Life Cycle Model. Then, the spiral model is compounded of three steps: Planning, Engineering, and Evaluation repeated through two iterations, as shown in Fig. 4.2. The first iteration was focused on reading the model database and it shows this in the CG web page. Meanwhile, in the second iteration, the main aim was to create a zip file with the application and server code ready to run on the development environment. In the following sections, the stages of each iteration of the Spiral Model are described.

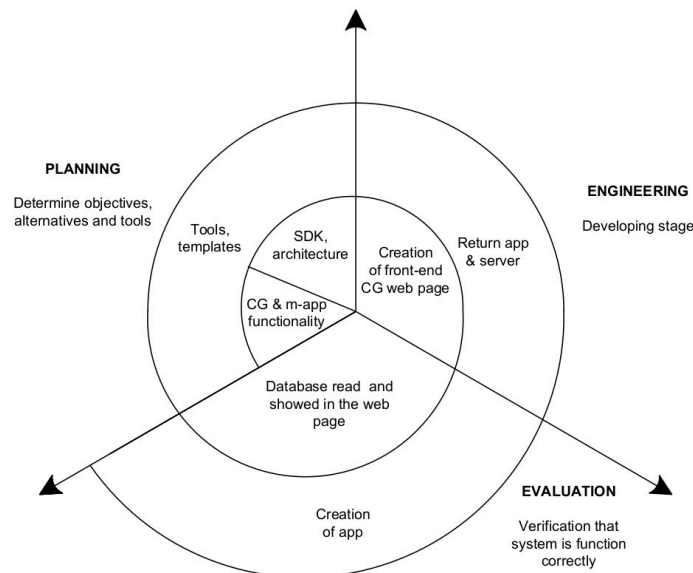


Figure 4.2: Spiral model for a CG.

4.1 Planning

This stage consists of determining the functionality of the CG and mobile application, SDK, architectural pattern selection, hardware and software, front-end characteristic, templates, and back-end characteristic of the CG and mobile application. The main activities related to each iteration are shown in Table 4.1.

Iteration	Planning
First Iteration	-Definition of CG functionality -Definition of mobile application functionality -SDK selection -Architectural pattern selection
Second Iteration	-Templates definition for mobile application interface and code -Tools definition for CG and mobile applications -Implementation Constrains given by Selected Tools

Table 4.1: Spiral Model planning summary to create a CG for mobile applications.

4.1.1 Definition of CG Functionality

In the last times, mobile applications have demonstrated that 80% of their code is focused on managing data through windows to insert, read, update, and delete information [1]. Therefore, to improve the productivity of developing a mobile application that is mainly focused on managing data, the functionalities of the CG web page was generated. Taking account the following constrains for mobile application data manage:

- Insert could be used with all variables of the database.
- Update is not used by primary or foreign keys variables.
- Delete option is a boolean variable that changes to true avoiding delete information from the database.
- Read functionality takes all information where delete database variable is false if these variables exist, else it takes all information.

Then, the CG front-end shall select an abstract data model that contains the information from the database system. This data shall be used to display each database table configuration

that may be a button in the mobile application. Therefore, the configuration of the mobile application consists of complete the following requirements that are summarized in Fig. 4.3:

- a) To select a database table that shall be represented as a button in the mobile application, writes its name, its description, to choose how will be ordered the table information and to select the variable used to delete data.
- b) To select variables of tables that will be shown in the mobile application, its label, and the variable may be inserted or used to search by the user of a mobile application.
- c) Complete database information to CG creates the server connection to read, insert, update, and delete data from the mobile application.

The screenshot shows a web-based configuration interface. At the top left, a dropdown menu is set to 'reader'. Below it, a form for database connection details is visible, with fields for 'Db User:', 'Password:', 'Ip Direction:', 'Port:', and 'Db Name:'. A 'FINISH ALL' button is located to the right of this form. Below the database form, there is a table configuration section. It has a header row with columns: 'Table Name', 'Show?', 'Label', 'Description', 'Order by', and 'Delete by'. The first row shows 'reader' as the table name, with a checked 'Show?' checkbox, a 'reader' label, an empty description, and two 'Select Table...' dropdowns for ordering and deleting. Below this is a variable configuration section with a header row: 'Variable Name', 'Show?', 'Label in app', 'Insert', 'Update', and 'Search'. The variables listed are 'id_reader (serial)(PK)', 'full_name (character varying(40))', 'phone (character varying(10))', and 'delete (boolean)'. Each variable has a 'Show?' checkbox and corresponding input fields for 'Label in app', 'Insert', 'Update', and 'Search'.

Figure 4.3: CG front-end used to configure mobile application settings.

Finally, after the user completes all mobile application settings, he could make click in the “FINISH ALL” button and obtain his mobile application.

4.1.2 Definition of Mobile Application Functionality

In the last years, the transactional applications are increased. In the case of the food industry, there are applications to obtain nutritional information of products, to keep accounts, to buy food, to take orders, and many more. Then, to calculate the cost of food recipes, a mobile application that allows to insert, read, update and delete shall be developed.

To produce a mobile application to determine the cost of recipes for the restaurant industry is necessary to calculate the indirect manufacturing cost (IMC), the direct raw material (DRM), taxes, and profitability. Then, the application needs to work with products, employees, devaluation of objects, and other costs that allow to obtain the final costs of recipes. Furthermore, an analysis of each recipe could be made with the calculation of gross profit and net profit. Therefore, the mobile application shall be constructed with a database that makes all calculations through triggers and functions.

4.1.3 SDK Selection

This stage aims to select a SDK appropriated to create mobile applications that can be used by the CG. According to the bibliographic investigation of Section 3.2 that is summarized in Table 3.2, the following items were taken into account to select an adequate SDK for being used by the CG:

- To avoid the use of third party plugins to access to system features.
- High graphical performance (high raw speed of images with a smoothness user interface).
- Use of a well-known/popular programming language.
- High modularization of code.
- Allows creating good and friendly user interfaces (intuitive, elegant, consistent, reliable).
- High compatibility with android and iOS applications and allows distributing the created software through application stores.
- Open source and free of charges.
- Instantaneously reload of mobile applications.
- Decreasing security problems avoiding the use of HTML5 or third party plugins.

After literature exploration, Flutter demonstrated meet most features mentioned before. It has a good graphical performance, high modularization level, it is free, it is open-source, and

it is more secure than other cross-platform SDKs because it uses little third party plugins and does not use HTML5 that is more insecure than other programming languages. Then, with data obtained was possible to conclude that Flutter will be the SDK to create mobile applications.

4.1.4 Architectural Pattern Selection

In base to bibliographic revision of the Section 3.2.3 the MVC architecture was determined for the CG and mobile applications. MVC architecture satisfies the conditions described in Table 4.2 either for CG and mobile applications.

MVC features	CG	Mobile applications
(+) Facilitate the testing stage	✓	✓
(+) Facilitate maintainability	✓	✓
(+) Facilitate scalability	✓	✓
(+) Components are reusable	✓	✓
(+) Compatible with big development groups	✓	✓
(-) It is necessary create more components	✓	✓
(-) It is necessary maintain more component	✓	✓

Table 4.2: Architectural pattern features for CG and mobile applications.

Based on MVC architecture, two views were planned to the CG web page as is shown in the Fig 4.4. This figure shows how the view is divided into two web pages. The first, homepage-Setting.jsp transfers the information to the model that reads all the information, and it creates a database object that is shown in the tableSetting.jsp. Furthermore, this second view detects mobile application requirements that will be used by the model to create the final application.

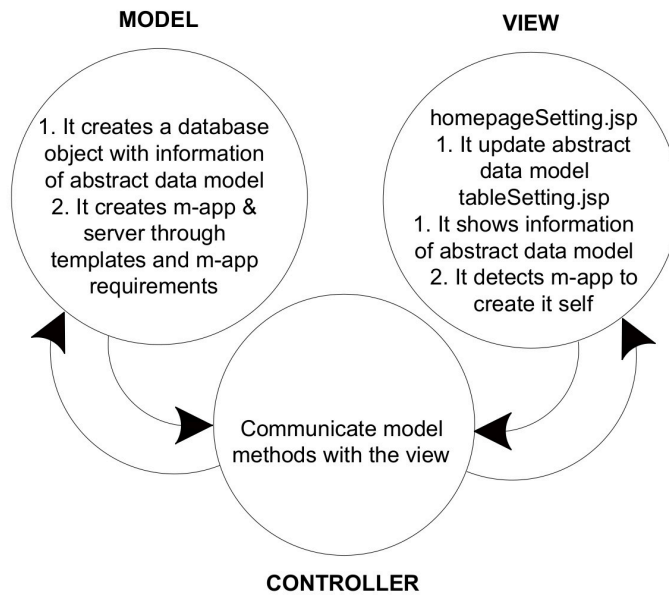


Figure 4.4: MVC architecture for the CG

On the other hand, the MVC architectural pattern for the mobile application was distributed, according to Fig. 4.5 that describes how the view reads all information after make the insert, read, update, and delete operations that will be executed by the model.

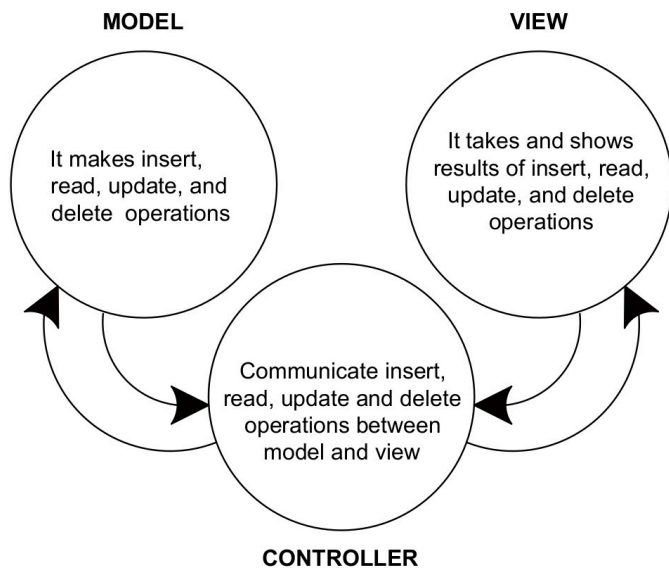


Figure 4.5: MVC for mobile applications developed with the CG

4.1.5 Templates definition for Mobile Application Interface and Code

This stage contemplates the definition of the mobile application user interface that will be created by the CG. Therefore, a standard user interface is created to build the templates based on the definitions summarized in the Fig.4.6, Fig. 4.7, and the Fig. 4.8:

- **DEF 1:** The main window of the application will allow buttons with titles and subtitles. The subtitles will be optional to insert.
- **DEF 2:** Each table of the user database could have one window in the application that will be opened from one button of the main window.
- **DEF 3:** The windows created in the application through the database will have a search and checkbox to delete information. Furthermore, this window will have one button to delete, one button to update, and one button to return at the main window of the application.

a)

Table Name	Show?	Label	Description	Order by	Delete by
reader	<input checked="" type="checkbox"/>	reader		Select Table...	Select Table...
Variable Name	Show?	Label in app	Insert	Update	Search
id_reader (serial)(PK)	<input type="checkbox"/>	id_reader	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
full_name (character varying(40))	<input type="checkbox"/>	full_name	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
phone (character varying(10))	<input type="checkbox"/>	DEF_3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
delete	<input type="checkbox"/>	DEF_1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

b)

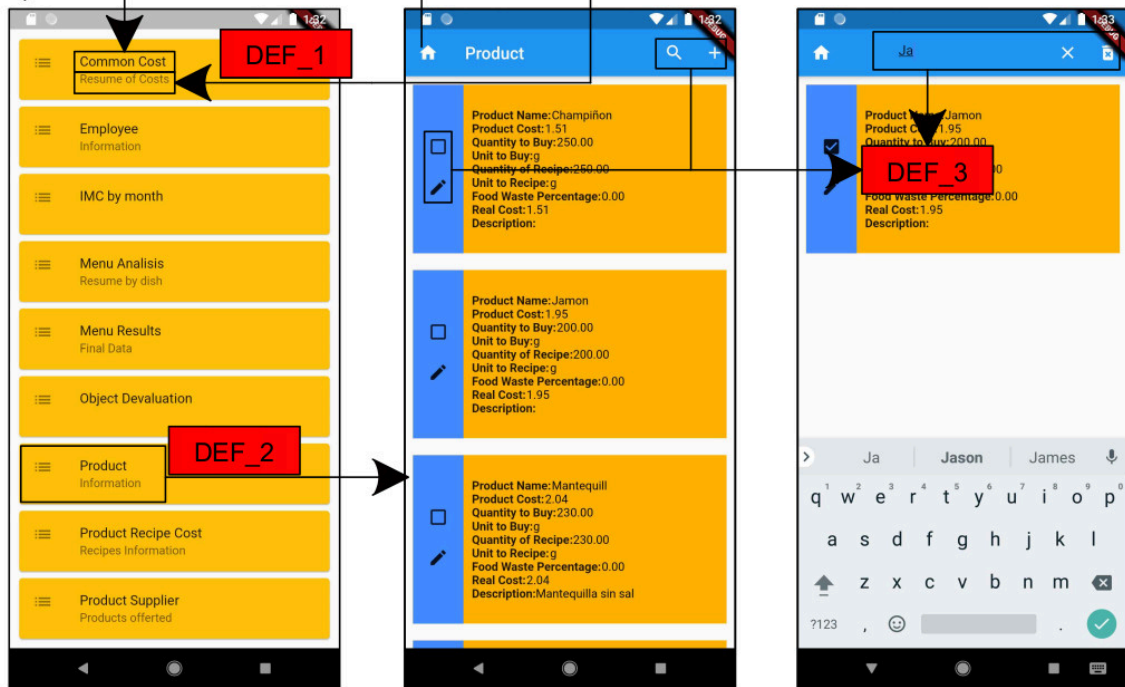


Figure 4.6: Design of a Standard UI based on definition DEF_1-DEF_3. a) tablePageSetting.jsp and b) Mobile application.

- **DEF_4:** The delete and update button will be the same in the application to all windows created through the tables of the database. Then, this will be complete dynamically.



Figure 4.7: Standard update and insert windows for mobile applications based on DEF_4 .

- **DEF_5:** The foreign keys attributes of the application will be buttons with the information of their primary keys variables.
- **DEF_6:** Users will not be confused with “primary keys ids” in the application. Then, foreign key buttons will be shown information related to the primary key button.
- **DEF_7:** The update, delete, and insert button will have a dialog box to confirm the user requirement.

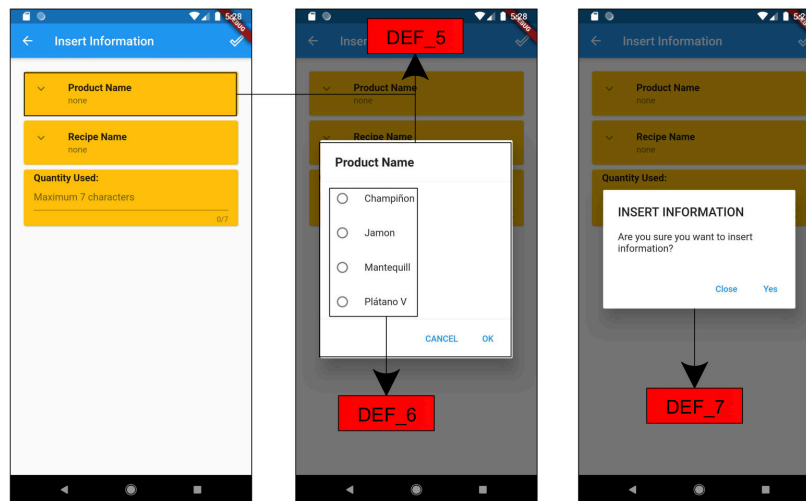


Figure 4.8: Standard manage of primary and foreign keys based DEF.5 and DEF.6. Furthermore, standard confirmation message before to make an insertion in the database from mobile application based on DEF.7.

After defining the standard mobile user interface that will produce the CG, the templates for complete software architecture of the mobile applications code are established and used posterity in the implementation. The necessary templates contain tags bounded by “<< >>”. For instance, << *dbName* >> will be replaced with the name of the database inserted by the user when the code is generated. PHP and Flutter templates were created after defined its tools and posterior used through writeTemplate Java EE method of the class genTemplate that receives the path of the template, the path of the user code, the words that will be changed, and the new words to create a new file.

- **PHP templates:** These templates allow connecting the Apache server with the database and mobile application. For instance, the code of the dbInformation.php shown in the Algorithm 4.1 is a template that contains the information necessary to connect the database with the server as ip, port, database name, user name, and password.

```

1 <?PHP
2 $conexion = pg_connect("host=<<ip>> port=<<port>> dbname=<<dbName>> user=<<dbUser>>
   password=<<password>>")
3     or die('Lost Connection: ' . pg_last_error ());
4 ?>
```

Algorithm 4.1: PHP template to connect database with apache server.

This template will be replaced by the information of the userDb object of the Algorithm 4.2 through writeTemplate Java EE method.

```

1 // Creation dbInformation.php
2 ArrayList<String> wordsToChange = new ArrayList();
3 wordsToChange.add("<<dbUser>>");
4 wordsToChange.add("<<password>>");
5 wordsToChange.add("<<ip>>");
6 wordsToChange.add("<<port>>");
7 wordsToChange.add("<<dbName>>");
8 ArrayList<String> newWords = new ArrayList();
9 newWords.add(userDb.getDbUser());
10 newWords.add(userDb.getPassword());
11 newWords.add(userDb.getIpDirection ());
12 newWords.add(userDb.getPort());
13 newWords.add(userDb.getName());
14 genTemplate.writeTemplate(pathToWrite, pathToCopy, wordsToChange, newWords);
```

Algorithm 4.2: Java code to replace PHP template of the Algorithm 4.1.

- **Flutter template:** It is a template of Flutter project that has tags to create mobile application according to user requirements. For instance, the tags `<< appDescription >>`, `<< appNames >>`, and `<< listOfScreen >>` are used to create the buttons of the mobile application main window. This tags are shown in the Algorithm 4.3.

```

1 // Creation list of screens
2 class _Routes extends State<DynScreens> {
3   //Global variables
4   List appDescription = [<<appDescription>>];
5   List appScreenNames = [<<appNames>>];
6   List listOfScreen = [<<listOfScreen>>];
7   @override
8   //View of principal screen
9   Widget build (BuildContext context) {
10    return Scaffold (
11      // List of widges
12      body: Center(
13        child: Container(
14          padding: EdgeInsets.all (8.0) ,
15          child: ListView.builder (
16            itemCount: appScreenNames.length,
17            itemBuilder: (BuildContext ctxt , int i) {
18              return Column(
19                children : <Widget>[
20                  //Button to each screen
21                  Card(
22                    color: Colors.amber,
23                    child: ListTile (
24                      title : Text(appScreenNames[i]),
25                      subtitle : Text( appDescription [ i ] ) ,
26                      leading: const Icon(Icons.list) ,
27                      onTap: () {
28                        Navigator.push(
29                          context ,
30                          MaterialPageRoute( builder : (context) => listOfScreen[ i ] ) ,
31                      );
32                    },
33                  ),
34                ),

```

```

35         ],
36     );
37     });
38     ),
39     ),
40     );
41 }
42 }

```

Algorithm 4.3: Flutter template to create buttons of the main window of the mobile application.

This template will be replaced with user information inserted in the CG through the use of the writeTemplate Java EE method as shown Algorithm 4.4..

```

1 // Creation of main window buttons
2 ArrayList<String> wordsToChange = new ArrayList();
3 wordsToChange.add("<<appDescription>>");
4 wordsToChange.add("<<appNames>>");
5 wordsToChange.add("<<listOfScreen>>");
6 ArrayList<String> newWords = new ArrayList();
7 newWords.add(userDb.getAppDescription());
8 newWords.add(userDb.getAppNames());
9 newWords.add(userDb.getListOfScreen());
10 genTemplate.writeTemplate(pathToWrite, pathToCopy, wordsToChange, newWords);

```

Algorithm 4.4: Java code to replace Flutter template of the Algorithm 4.3.

These templates need a copy inside a folder that shall be given to the user. This process is carried after the user makes click on the finish button. Then, to generate the zip file, some java packages were used inside JavaEE as “java.io” to read templates, “java.lang.Runtime” to execute Linux commands as copy templates, and “java.util.zip” to generate the zip file.

4.1.6 Abstract data model: File format

The model is an essential part of the process of code generation because it gives the information abstracted of the application into a graphical or textual language. Then, to select an abstract data model was taken into account the following parameter:

- The user shall transform his database into a model.
- The programming languages to create the CG shall read the model.

Therefore, XML was chosen due it is a textual language that is used for some programming languages (it is known) and satisfies the previous requirements.

4.1.7 Implementation Constrains given by Selected Tools

To create CG was essential a programming language to develop the CG web page, an abstract data model (defined previously), and a database. The web page was chosen due could be used in different operative systems that need fewer requirements than a server that could run the CG, and it does not require a previous installation. Then, Java EE, Postgres were selected due characteristics of the Table 4.3.

Java EE [84]	Postgres [85]
-It works with MVC	-It is free
-It allows to create a web page	-CG web page developer knows this database
-The Java language is the most popular according to TIOBE list	-It is open-source
-Java has a big community that helps to resolve problems	
-Java is known by the developer of this thesis	

Table 4.3: Tools definition for CG and mobile applications.

4.1.8 Constrains to Implement the CG and Mobile applications

Given the features by selected tools for development in Section 4.1.3, Section 4.1.7, CG and Mobile applications can be implemented under the following tool: Apache server that was chose to connect the database with the mobile application due works with PHP [86] language that is very known in the world and by the CG developer. On the other hand, hardware and software

applications given by selected tools were established to run the CG and mobile application in the server available. This information is summarized in the following items:

- The CG could run in a server of a 64-bit architecture.
- The CG will be run in a Linux Mint 18.1 server with 8 CPUs.
- The CG will be using a NetBeans IDE 8.2 with Java EE 8, JDK 1.8.0_222, and Apache Tomcat 9.0.12.
- The application will run in Flutter 1.7.8+hotfix.4.
- The Dart version used to program will be 2.4.0.
- The application will be proved in the Flutter emulator with Android version greater or equal than 4.1 and API 16.

4.2 Engineering

This stage includes tasks related to the design and development of the front-end of the CG for the first iteration and the back-end of CG for the second iteration.

4.2.1 Design

The relevant design diagrams to develop the CG are the following:

Package Diagram

The CG application is compounded by three main packets as is depicted in Fig. 4.9. These packages are Controller, Model, and “codeGenerator/web” folder that contains the view.

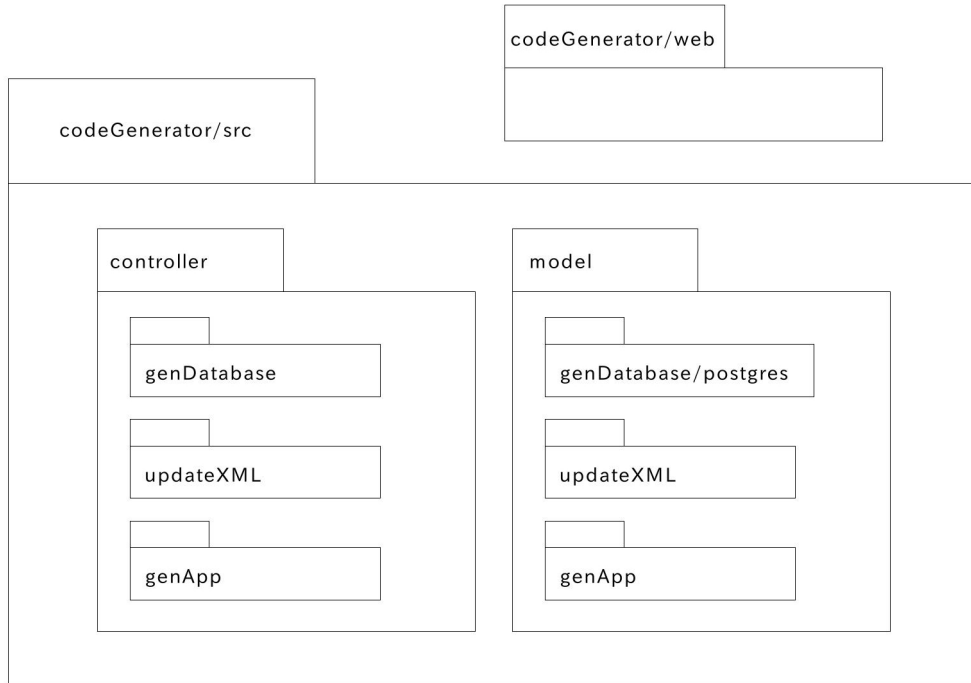


Figure 4.9: Package diagram of the CG

Moreover, the mobile application code is organized inside the lib package (folder used in Flutter mobile applications to develop applications) of the mobile application template. It is shown in the package diagram of Fig. 4.10 where the main.dart is the main window that contains all table buttons set in the CG. The insert.dart and update.dart files are dynamic windows used by the screen.dart that changes its name according to user requirements. Furthermore, these files were copied according to the number of windows that were selected by the user.

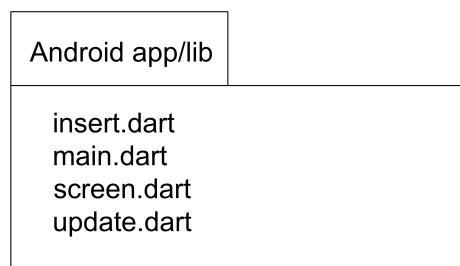


Figure 4.10: Package diagram of the mobile applications.

Deployment Diagram

The CG will run in a server, and the clients will access the CG application through a web browser. In this sense, the CG application will be deployed as is shown in Fig. 4.11.

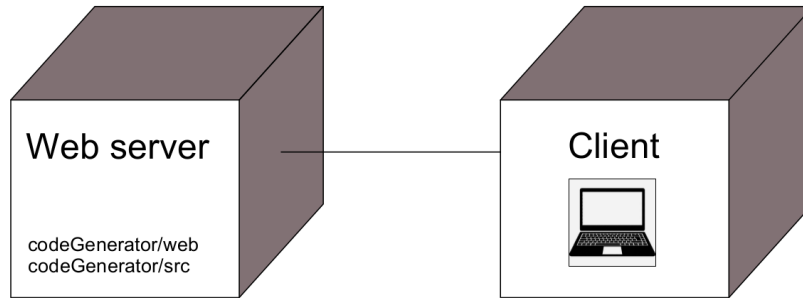


Figure 4.11: Deployment diagram of CG.

Furthermore, the mobile application will connect to the database server through web service, as is illustrated in Fig. 4.12

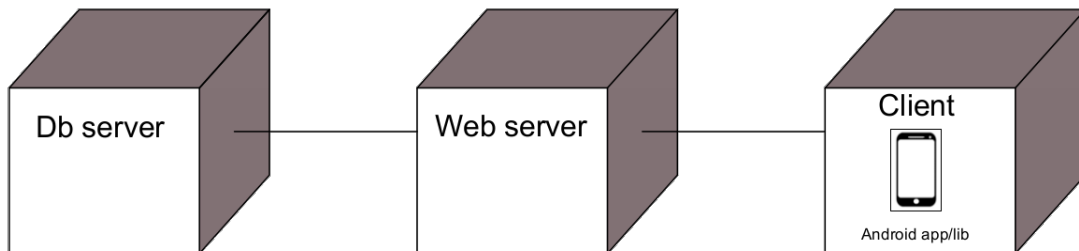


Figure 4.12: Deployment diagram of mobile application.

4.2.2 Implementation

It is focused on writing the CG code based on the software specification and using the software tools selected on Section 4.1.7 taking account tools of the Section 4.1.8 to obtain code that is replaced according user requirements as the examples of Algorithm 4.1 and Algorithm 4.3.

4.3 Evaluation

This stage allows to determine the quality of both applications: CG and generated mobile application. This project measures the quality in terms of the functionality and performance metrics as presents Chapter 5 Section 5.1 and Section 5.2.

In this stage, the first iteration aims to evaluate the front-end of CG. Meanwhile, for the second iteration, evaluates the quality of the generated mobile application application (For evaluation purposes, this project creates a mobile application to calculate the recipe cost of restaurants), after deploying the zip file produced from CG application.

All experiments and obtained results for CG and mobile application applications are described in Chapter 5 Section 5.3 and Chapter 6, respectively.

Chapter 5

Experimental Setup

Software quality metrics allow measuring software performance, usability, productivity, and many other uses. Quality metrics described in 5.1 and 5.2 are used, on the experiments of the Section 5.3, to evaluate the quality of the CG and mobile application, respectively.

5.1 CG Metrics

This section introduces metrics evaluated in posterior experiments to determine the quality of the CG.

5.1.1 CG Functionality

It aims to determine what requirement are satisfied in concordance with Section 4.1.1. Therefore, every requirement that is satisfied has the same weight as shown Table 5.1.

Metrics	Description
Model updated	The database model is updated and represented in the CG front-end
User settings	CG allows selecting each table of the database to create buttons according to user settings
Application generated	A zip file is returned with the server and mobile application

Table 5.1: Metrics for functionality of the CG.

5.1.2 Performance Metrics of CG Web Page

It refers to the velocity of the web page based on several parameters like time to display the content, time to appear first image, or text, among others. All parameters related to performance are shown in Table 5.2. They have different weights according to heuristics analysis of the Google Team [87]. If the final punctuation is over 90%, means deployed web application is fast.

Metrics	Description	Weighting
First Contentful Paint	It is the time used to appear first image or text	20.0%
Speed Index	It shows the velocity to display the content of the web page	6.7%
Time to Interactive	It is the time used to make a full interaction	26.7%
First Meaningful Paint	It is the time used to show the main content	33.3%
First CPU Idle	It indicates when the main thread has a minimally interactive	13.3%

Table 5.2: Weights for “tableSetting.jsp” performance.

5.1.3 Accessibility Metrics of CG Web Page

It refers to the usability through metrics as an accessible button, contrast ratio, title, id attributes, labels, and others. This parameters are detailed in the Table 5.3. They have a different weight according to the accessibility function that satisfies. Its weight is given by accessibility audits of the Google Team [87]. Finally, a grade over 90% implies that the web page has high accessibility.

Metrics	Description	Weighting
Accessible button	It is not necessary to write the word “button” because the user can recognize each button	4.7%
Contrast ratio	The contrast between the background and words allows reading the text	1.4%
Title	The title gives an overview of the web page	1.4%
id attributes	The id is different for each attribute	0.5%
lang attributes	It contains the “lang” attribute that indicates the language of the web page	1.4%
Labels	The attributes have their labels	4.7%
Others	It refers to parameters that could no be analyzed in the CG because these metrics do not apply. Therefore, the “others” metric gives the score of the table automatically	85.9%

Table 5.3: Weights for “tableSetting.jsp” accessibility.

5.1.4 Best Practices Programming Metrics of CG Web Page

It means good programming practices. It works with variables as geolocation, detect JavaScript libraries, errors logged, among others. Its attributes are localized in Table 5.4. The weight of each attribute is obtained, dividing 100% by the number of attributes. A grade greater or equal than 90% implies that web application has high standards of programming.

Metrics		Description	Weighting
Avoid Cache		The web page does not use cache	6.66%
Https		The web page needs the protocol Http to be a secure web page	6.66%
Http/2		The web page uses the protocol Http/2 that is better than Http/1.1	6.66%
Passive listeners		The web page does not use passive listeners because they delay the scroll of the web	6.66%
Avoid document.write()	docu-	The web page does not use external scrip calls via document.write() because delay loads the web page	6.66%
Cross-origen	destina-	The web page add rel="noopener" or rel="noreferrer" to external links avoiding vulnerabilities and improving the performance	6.66%
Geolocation		The web page does not give a request of location without motive	6.66%
HTML doctype		The web uses the attribute "doctype". This web prevents that web page could switch to "Quirks Mode" to be compatible with old web browsers	6.66%
Avoid scripts	third-party	The use of third-party scripts could produce vulnerabilities; for this reason, it is better not to use this type of scripts	6.66%
Detect libraries	JavaScript	The browser can detect the JavaScript libraries used	6.66%
Avoid request notification		The web does not give notification without a context to the user	6.66%
Avoid deprecated APIS		The web page does not use APIS that is deprecated	6.66%
Paste password		The web page does not allow to paste the password by security	6.66%
Errors logged		The error message does not appear in the console where the user is logged	6.66%
Images		The web allows displaying an image with a correct aspect ratio	6.66%

Table 5.4: Weights for "tableSetting.jsp" best programming practices.

5.1.5 SEO Metrics of CG Web Page

It represents the facility of search engines to find the CG. The variables analyzed for these metrics are viewport, document title, descriptive text, among others. Its attributes related to SEO are in Table 5.5. Each variable has the same weight, and grades greater or equal than 90% represent that web application could find easier by search engines like Google Search or Yahoo.

Metrics	Description	Weighting
Viewport	It is configured as a viewport for mobile devices	7.7%
Document title	The web title gives an overview of the web page	7.7%
Meta Description	The viewport for mobile devices has a meta name to optimize its screens	7.7%
Http status	The code of web page status is given correctly	7.7%
Descriptive text	The text used is descriptive so that search engines can understand the content	7.7%
Active for search engines	The web page is not blocked to search engines	7.7%
Language	The system engines can understand the language used on the web page because the language is declared	7.7%
Avoid plugins	Some system engines cannot understand the plugins; for this reason, it is better not using them	7.7%
Meta description	The web page has a summary of the page content	7.7%
Targets	Interactive elements should be more significant than 48x48 px and need to have enough space among them	7.7%
Others	It refers to parameters that could no be analyzed in the CG because these metrics do not apply. Therefore, the “others” metric gives the score of the table automatically	23%

Table 5.5: Weights for “tableSetting.jsp” SEO.

5.2 Application Metrics

This section explains metrics that will be used in posterior experiments by the mobile application.

5.2.1 Application Functionality

It aims to determine what requirements are satisfied for mobile application in concordance with functionalities of the Section 4.1.2. Therefore, every requirement that is satisfied has the same weight as shown Table 5.6.

Metrics	Description
Main window	The main window contains buttons to represent the information of the database
Variables information	Every button of the main window has another window to represent its information
Transactional operations	Transactional operations are made correctly
Box of dialog	Insert, update, and delete buttons show a dialog box to confirm or cancel the operation
Foreign keys buttons	Foreign keys buttons show information of its primary keys
Order and search	Information is ordered and searched according to user specifications

Table 5.6: Metrics for functionality of the mobile application.

5.2.2 Application Compatibility

It was constructed to android mobile devices with versions of the operative system greater or equal than 4.1 and with an API greater or equal than 16. Therefore, every device used to analyze the compatibility of the mobile application obtained by the CG has the same weight as shown Table 5.7.

Metrics	Description
Android Jelly Bean	Mobile with android Jelly Bean 4.1, API 16 and resolution of 320x480 mdpi
Android Lollipop	Mobile with android Lollipop 5.1, API 16 and resolution of 240x320 mdpi
Android Nougat	Mobile with android Nougat 7.1.1, API 19 and resolution of 480x854 hdpi
Android Oreo	Mobile with android Oreo 8.1, API 27 and resolution of 1080x2220 xxhdpi
Android Pie	Mobile with android Pie 9.0, API 22 and resolution of 720x1280 xhdpi
Android Q	Mobile with android Q, API 29 and resolution of 1440x2960 xxhdpi

Table 5.7: Metrics for compatibility of the mobile application.

This work performs two experiment sets to measure the quality of the CG: a) CG experiments to evaluate the capability of the CG to configure the insert, read, update, and delete operations of a given abstract data model. b) Mobile application experiments to evaluate the quality of the mobile application generated by the CG.

5.3 CG Experiments

For experimental purposes and based on deployment diagram of Fig. 4.11, the current project deploys the CG application on a Laptop that works with tools of Section 4.1.8. From client computer, the user access at “<http://192.168.1.2:8082/fluttercode.com/>” to start the CG application. Resource details for these experiments are depicted in Figure 5.8.

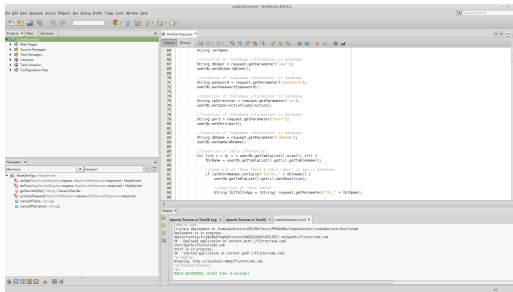
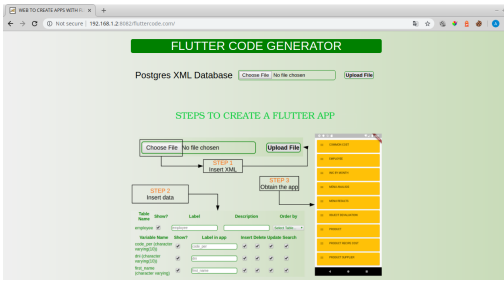
Server Computer	Client Computer
	
<ul style="list-style-type: none"> • Operative System: Linux Mint 18.1 • Processor: Intel Core i7-8550U • Graphic card: AMD Radeon 530 4GB GPU • Monitor: 15.6” • Storage: 2 TB 	<ul style="list-style-type: none"> • Operative System: Linux Mint 18.1 • Processor: AMD Dual-Core E-450 APU/1.65 GHz • Graphic card: AMD Radeon HD 6320 • Monitor: 11.6” • Storage: 500 GB

Table 5.8: Server and Client used for CG experiments

5.3.1 Experiment 1: Functionality of the CG.

Based on CG user requirements (see Section 4.1.1), this experiment mainly evaluates the ability of the CG to: 1) read an XML file (it contains the definition of the data base for determining the recipe cost of a restaurant), 2) configure the behaviour of the insert, read, update, and delete operations on the mobile application, and 3) create the mobile application based on established templates.

5.3.2 Experiment 2: Analysis of performance, accessibility, best practices programming, and SEO metrics of the CG through Lighthouse tool.

In this experiment, the client installs the Lighthouse tool extension in Google Chrome. After, the user opens the website <http://192.168.1.2:8082/fluttercode.com/> and update the abstract data

model. Finally, the user makes click in the button named “generate report” of the Lighthouse tool.

5.4 Mobil Application Experiments

This section shows the experiments used to test the mobile application developed inside an internal network, as shows in the deployment diagram of Fig. 4.12.

5.4.1 Experiment 1: Functionality of the mobile application.

This experiment is done through the use of the zip file obtained from Section 4.2. Then, the user needs to run the Postgres server and Apache server. Furthermore, the user needs to install the application into a mobile device and prove application functionality metrics of Section 4.1.2 are satisfied.

5.4.2 Experiment 2: Compatibility of the mobile application.

In this experiment, the user needs to use the recipes cost mobile application of Section 4.1.2. Then, the user runs the Postgres and Apache server to proof the functionalities of this table in each mobile device of the Table 5.7.

Chapter 6

Results

A CG in Java EE (Object-Oriented programming) with a MVC Architecture Pattern was created. The CG produces a Flutter application that allows making the insert, read, update, and delete operations. The process of creating an application initializes with the developer that inserts an XML Postgres database to the homePage.jsp; after, the database information is showed in the tablePage.jsp to the developer. Then, the developer completes all the information required to produce the Flutter app, and finally, the CG returns a zip file with the mobile application as shown Fig. 6.1.

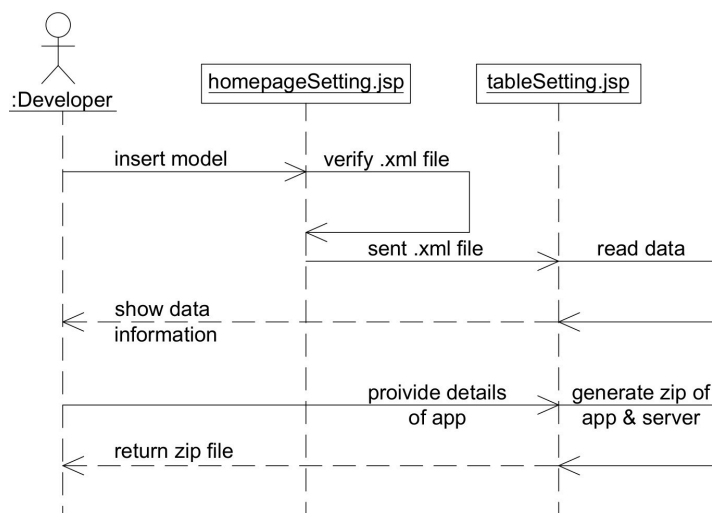


Figure 6.1: Sequence diagram to create an application with the CG developed.

This Chapter details the results obtained from experiments of Chapter 5. Then, CG results and application results have been detailed in the following sub-sections.

6.1 CG Results

Analysis and interpretations of grades obtained by the CG in the experiments of Section 5.3.

6.1.1 Experiment 1: Functionality of the CG.

The CG functionality metrics of Section 5.1.1 are evaluated in this experiment. The results obtained from Table 6.1 implies that all CG metrics are satisfied. This result is benefiting because it means that CG works according to CG functionality of Section 4.1.1. Then, it is not necessary to make functionality changes.

Metrics	Satisfies?
Model updated	✓
User settings	✓
Application generated	✓

Table 6.1: Results of CG, experiment 1: Functionality.

6.1.2 Analysis of performance, accessibility, best practices programming, and SEO metrics of the CG through Lighthouse tool.

The Lighthouse tool helps to identify the quality aspect that could be improved on the web page. Grades of 100% imply that all quality aspects are satisfied; meanwhile, 0% means that there are not quality aspects satisfied. The grades obtained where 100% to performance, 100% to accessibility, 86.68% to best practices of programming, and 100% to SEO of CG web page. Then, on average, the 96.67% of CG quality aspect is satisfied. However, the CG could implement Https and Http/2 protocols to have 100%. The result of 96.67% is good because the Lighthouse tool gives grades lower than 80% in the majority of the web pages, for example, the CG Flutter Studio has a grade of 51% in performance, 54% in accessibility, 100% in best practices and

76% in SEO that gives an average of 70.25%. Therefore, it is possible to say that CG develop has a high quality aspect.

Performance of CG Web Page

According to Table 6.2, all the parameters that have a relation with the velocity of the web page are satisfied. Then, the time used for the developer to create the application is not increased with CG web site uploads. Therefore, the user obtains as a benefit that he does not need to lost time until the website is upload.

Metrics	Satisfy?
First Contentful Paint	✓
Speed Index	✓
Time to Interactive	✓
First Meaningful Paint	✓
First CPU Idle	✓
TOTAL	100%

Table 6.2: Results of CG, experiment 2: Performance.

Accessibility of CG Web Page

Table 6.3 indicates that parameters that improve user usability are satisfied by the web site. Then, the developers could manipulate the CG web site of a satisfactory form to create mobile applications because he has the benefit that could understand and manage the CG without comprehension problems.

Metrics	Satisfy?
Accessible button	✓
Contrast ratio	✓
id attributes	✓
lang attributes	✓
Labels	✓
Others	✓
TOTAL	100%

Table 6.3: Results of CG, experiment 2: Accessibility.

Best Practices Programming of CG Web Page

The most parameters of best practices are satisfied by the CG web page. However, the CG needs to implement Https and Http/2 protocols to be more secure as the Table 6.4 shows. Therefore, the webpage could be intercepted by a third person to read the model or the application obtained by the CG. Then, it is recommended using this CG into an internal network or implement the Https security protocols to work with this on the internet.

Metrics	Satisfy?
Avoid Cache	✓
Https	X
Http/2	X
Passive listeners	✓
Avoid document.write()	✓
Cross-origen destination	✓
Geolocation	✓
HTML doctype	✓
Avoid third-party scripts	✓
Detect JavaScript libraries	✓
Avoid request notification	✓
Avoid deprecated APIs	✓
Paste password	✓
Errors logged	✓
Images	✓
TOTAL	86.68%

Table 6.4: Results of CG, experiment 2: Best practices programming.

SEO of CG Web Page

The CG could be easy to find in internet search engines by developers if it is published because satisfying all the SEO parameters of the Lighthouse tool [87]. This result (see Table 6.5) is beneficial in the case that the CG will be published on the internet because this parameter facilitates to find the website on the search engine like Google Search or Yahoo.

Metrics	Satisfy?
Viewport	✓
Document title	✓
Meta Description	✓
Http status	✓
Descriptive text	✓
Active for search engines	✓
Language	✓
Avoid plugins	✓
Meta description	✓
Targets	✓
Others	✓
TOTAL	100%

Table 6.5: Results of CG, experiment 2: SEO.

6.2 Application Results

Analysis and interpretations of grades obtained by the mobile application in the experiments of Section 5.4.

6.2.1 Functionality of the mobile application

The mobile application is evaluated according to the metrics of Section 5.2.1. The results obtained in Table 6.6 imply that all metrics are satisfied. This result is beneficial because it means that CG created could produce applications without compilation problems where the developer not necessary will need to make a change to the code manually.

Metrics	Satisfy?
Main window	✓
Variables information	✓
Transactional operations	✓
Box of dialog	✓
Foreign keys buttons	✓
Order and search	✓

Table 6.6: Results of mobile application, experiment 1: Functionality.

6.2.2 Compatibility of the mobile application.

The compatibility of the mobile application is through the installation of mobile application in different operative systems to confirm that there are no errors caused by fragmentation problems. Therefore, according to Table 6.7, the result obtained is a benefit for the mobile application because this application could be installed in any device with an android greater or equal than Jelly Bean 4.1 and API 16. However, it could exist specific problems in some devices that were not be analyzed.

Metrics	Satisfy?
Mobile with android Jelly Bean 4.1, API 16 and resolution of 320x480 mdpi	✓
Mobile with android Lollipop 5.1, API 16 and resolution of 240x320 mdpi	✓
Mobile with android Nougat 7.1.1, API 19 and resolution of 480x854 hdpi	✓
Mobile with android Oreo 8.1, API 27 and resolution of 1080x2220 xxhdpi	✓
Mobile with android Pie 9.0, API 22 and resolution of 720x1280 xhdpi	✓
Mobile with android Q, API 29 and resolution of 1440x2960 xxhdpi	✓

Table 6.7: Results of mobile application, experiment 2: Compatibility.

Chapter 7

Conclusions and Future Work

In general terms, a web CG in a MVC architectural pattern that produces a mobile application with Flutter SDK was created. This CG was developed using the Spiral model, it satisfies all functionalities aspects and it obtains a grade of 96.67% with the “Lighthouse” Google tool that allows to obtain grades from 0% until 100% in the performance, accessibility, best practices, and SEO metrics. Furthermore, the mobile application obtained allows to manage the insert, read, update, and delete operations through an Apache server that has a connection with the application created by the CG according to user requirements. Then, the CG satisfies the following statements:

- The MVC architectural pattern was established for the CG and mobile applications because it facilitates the testing stage, maintainability, and scalability. Furthermore, its components are reusable and are compatible with large developed groups. On the other hand, this pattern needs to create and maintain more components.
- The hardware and software used to run the CG is a Linux Mint 18.1 operative system of 64 bits architecture that has installed NetBeans IDE 8.2 with Java EE 8, JDK 1.8.0 222, and Apache Tomcat 9.0.12. Inside this server, the templates of the mobile application are replaced and sent to the user of the CG.
- According to the information collected a good SDK to develop a CG is Flutter because it is more secure, it is open source, it is free, it does not work with Html5, JavaScript,

and it tries to restrict the use of third party plugins. Furthermore, it is multi-platform because Flutter allows to create Android and iOS mobile applications, and it has excellent performance.

- The CG satisfies all functionalities and the grade obtained were 100% to performance, 100% accessibility, 86.68% to best practices of programming, and 100% SEO metrics. Therefore, the quality aspect of the CG is 96.67%; however, could be improved if the Https and Http/2 protocols are implemented to have better programming practices.
- An application that calculates the cost of recipes for the restaurant industry was developed with the CG. This application satisfies all user and system functionalities. Moreover, the application approved all tests of compatibility. Then, the application could be installed in any device with android greater or equal than Jelly Bean 4.1, and API 16. However, the application needs be proved in iOS operative system greater or equal than 8 according with Flutter SDK documentation.

As future work, it is recommended to create new templates, incorporate new types, databases, and include security protocols. Furthermore, it will be possible to return code in different architectural patterns with different servers and SDKs to give more options to developers.

References

- [1] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice: Second Edition*, 2nd ed. Morgan & Claypool Publishers, 2017.
- [2] D. Grune, K. Van Reeuwijk, H. E. Bal, C. J. Jacobs, and K. Langendoen, *Modern compiler design*. Springer Science & Business Media, 2012.
- [3] M. Piefel, "A common metamodel for code generation," 2006.
- [4] S. Jörges, *Construction and evolution of code generators: A model-driven and service-oriented approach*. Springer, 2013, vol. 7747.
- [5] GeneXus. (1989) Crea y desarrolla soluciones de software sin precedentes, de forma automática. [Online]. Available: <https://www.genexus.com/es/>
- [6] C. O. T. LLC. (2008) Forms, reports, apps build and deploy rapidly. use offline, online, on-premises. [Online]. Available: <https://codeontime.com/>
- [7] e.World Technology Limited. (2002) Phpmaker 2020 = phpmaker x php report maker. [Online]. Available: <https://phpmaker.dev/index.php>
- [8] Y. Inc. (2000) Codecharge studio 5.1. [Online]. Available: <https://www.yessoftware.com/index2.php>
- [9] Scriptcase. (2000) Scriptcase 9.4. [Online]. Available: <https://www.scriptcase.net/>
- [10] SoftFluent. (2005) Softfluent codemodeler. [Online]. Available: <https://www.softfluent.com/product/codemodeler/>

- [11] R. G. G. Cattell, *Formalization and automatic derivation of code generators*. UMI Research Press Ann Arbor, Michigan, 1982.
- [12] B. Gonda and N. Jodal, “The genesis of genexus,” 2010. [Online]. Available: <https://www.genexus.com/en/global/company>
- [13] A. Bajovs, O. Nikiforova, and J. Sejans, “Code generation from uml model: State of the art and practical implications,” *Applied Computer Systems*, vol. 14, no. 1, pp. 9–18, 2013.
- [14] XLineSoft. (1999) Professional web apps with little or no coding. [Online]. Available: <https://xlinesoft.com/index.htm>
- [15] P. Chaisatien, K. Prutsachainimmit, and T. Tokuda, “Mobile mashup generator system for cooperative applications of different mobile devices,” in *International Conference on Web Engineering*. Springer, 2011, pp. 182–197.
- [16] H. Benouda, M. Azizi, R. Esbai, and M. Moussaoui, “Mda approach to automate code generation for mobile applications,” in *Mobile and Wireless Technologies 2016*. Springer, 2016, pp. 241–250.
- [17] M. Lachgar and A. Abdali, “Decision framework for mobile development methods,” *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 8, no. 2, 2017.
- [18] R. M. Gogonea, “Cordova generator: Generación automática de plugins apache cordova para aplicaciones híbridadas,” B.S. thesis, Universitat Politècnica de Catalunya, 2016.
- [19] I. Speed. (2003) Write 10,000 lines of code in 10 minutes! [Online]. Available: <http://www.ironspeed.com/products/Overview.aspx#gsc.tab=0>
- [20] F.-I. Ltd. (2004) dbqwiksite. [Online]. Available: <http://www.dbqwiksite.com/>
- [21] D. Gomez. (2009) Zathuracode generador de código. [Online]. Available: <https://zathuracode.org/>
- [22] S. N. Grösser, A. Reyes-Lecuona, and G. Granholm, *Dynamics of Long-Life Assets*, 2017.

- [23] L. D'Angelo, "Evaluation of code generation in agile software development of embedded systems," 2018.
- [24] W. M. Waite and G. Goos, *Compiler construction*. Springer Science & Business Media, 2012.
- [25] P. Mutisya. (2017) Flutter studio, version 2. [Online]. Available: <https://medium.com/@pmutisya/flutter-studio-version-2-41cce10fcf3d>
- [26] H. D. Gurad and V. Mahalle, "An approach to code generation from uml diagrams," *International Journal of Engineering Sciences & Research Technology*, vol. 3, no. 1, 2014.
- [27] N. Amanquah and S. Ndede, "Code generation on mobile devices for mobile apps," *Communications of the IIMA*, vol. 15, no. 2, p. 2, 2017.
- [28] B. Software. (2002) Create mobile-friendly web applications instantly without writing any code: It's easier, faster and much less expensive. [Online]. Available: <https://bigprof.com/appgini/>
- [29] SynApp2.org. (2008) Synapp2 is an ultra fast and effective web application builder. create beautifully integrated relational database management and reporting systems in record time. [Online]. Available: <http://www.synapp2.org/main/>
- [30] M. LLC. (2009) Deliver a perfect. [Online]. Available: <https://mobileroadie.com/>
- [31] AppMakr. (2010) Exciting changes are coming to appmakr. [Online]. Available: <http://www.appmakr.com/>
- [32] I. Shoutem. (2010) Mobile app creator. [Online]. Available: <http://www.shoutem.com/>
- [33] AppMachine. (2011) Create your own app or become a reseller and build apps for others. [Online]. Available: <https://www.appmachine.com/>
- [34] GoodBarber. (2011) Apps nativas para ios android. [Online]. Available: <https://es.goodbarber.com/>

- [35] T. Ltd. (2011) Mobile apps to connect with hard-to-reach people, really easily. [Online]. Available: <https://www.theappbuilder.com/>
- [36] Exadel. (2013) Everything you need to make your own app. [Online]. Available: <https://appery.io/>
- [37] A. Pie. (2013) App maker to make an app without coding. [Online]. Available: <https://www.appypie.com/>
- [38] B. Corp. (2013) App making made simple. [Online]. Available: <https://backendless.com/>
- [39] C. LLC. (2014) Codebhagat a unique, intelligent and real code generator for .net! [Online]. Available: <https://www.codebhagat.com/Store/BrowseProducts>
- [40] P. SpA. (2015) Create and publish powerful mobile apps no coding required. [Online]. Available: <http://www.apps-builder.com/privacy>
- [41] F. S. Franco Hernández. (2016) Anchurus-gen: generador de código php a partir de modelos isml. [Online]. Available: <https://repository.javeriana.edu.co/handle/10554/19571>
- [42] R. L. Glass, *Software Engineering: Facts and Fallacies*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.
- [43] Y. Zhang and S. Patel, “Agile model-driven development in practice,” *IEEE software*, vol. 28, no. 2, pp. 84–91, 2010.
- [44] J. Whittle, J. Hutchinson, and M. Rouncefield, “The state of practice in model-driven engineering,” *IEEE Software*, vol. 31, no. 3, pp. 79–85, May 2014.
- [45] G. Liebel, N. Marko, M. Tichy, A. Leitner, and J. Hansson, “Assessing the state-of-practice of model-based engineering in the embedded systems domain,” in *International Conference on Model Driven Engineering Languages and Systems*. Springer, 2014, pp. 166–182.

- [46] M. R. Chaudron, W. Heijstek, and A. Nugroho, “How effective is uml modeling?” *Software & Systems Modeling*, vol. 11, no. 4, pp. 571–580, 2012.
- [47] K. Baktha, “Mobile application development: All the steps and guidelines for successful creation of mobile app: Case study,” *International Journal of Computer Science and Mobile Computing*, vol. 6, no. 9, pp. 15–20, 2017.
- [48] M. Sarwar and T. R. Soomro, “Impact of smartphone’s on society,” *European journal of scientific research*, vol. 98, no. 2, pp. 216–226, 2013.
- [49] H. Wang, Z. Liu, Y. Guo, X. Chen, M. Zhang, G. Xu, and J. Hong, “An explorative study of the mobile app ecosystem from app developers’ perspective,” in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 163–172.
- [50] J. Clement. (2019) Mobile app usage - statistics facts. [Online]. Available: <https://www.statista.com/topics/1002/mobile-app-usage/>
- [51] D. Van Thanh and I. Jorstad, “The mobile phone: Its evolution from a communication device to a universal companion,” *TELEKTRONIKK*, vol. 101, no. 3/4, p. 3, 2005.
- [52] H.-J. Richstein and M. Kruempelmann, “Interactive software development kit documentation tool,” Jan. 22 2019, uS Patent 10,185,556.
- [53] S. Amatya and A. Kurti, “Cross-platform mobile development: challenges and opportunities,” in *International Conference on ICT Innovations*. Springer, 2013, pp. 219–229.
- [54] H. Heitkötter, S. Hanschke, and T. A. Majchrzak, “Evaluating cross-platform development approaches for mobile applications,” in *International Conference on Web Information Systems and Technologies*. Springer, 2012, pp. 120–138.
- [55] A. Biørn-Hansen, T.-M. Grønli, and G. Ghinea, “Animations in cross-platform mobile applications: An evaluation of tools, metrics and performance,” *Sensors*, vol. 19, no. 9, p. 2081, 2019.

- [56] Flutter. (2018) Technical overview. [Online]. Available: <https://flutter.io/technical-protect\discretionary{\char\hyphenchar\font}{}{}overview/>
- [57] B. anDRe ChaRLanD and B. Leroux, “Mobile application development: web vs. native,” *Communications of the ACM*, vol. 54, no. 5, 2011.
- [58] W. Jobe, “Native apps vs. mobile web apps.” *International Journal of Interactive Mobile Technologies*, vol. 7, no. 4, 2013.
- [59] K. H. Liu, “A taxonomy and business analysis for mobile web applications,” Ph.D. dissertation, Citeseer, 2009.
- [60] J.-P. Erkkilä, “Web and native technologies in mobile application development; web- ja natiiviteknologiat mobiilisovellusten kehityksessä,” G2 Pro gradu, diplomityö, 2013. [Online]. Available: <http://urn.fi/URN:NBN:fi:aalto-201304261933>
- [61] D. VESELÝ, “Analysis and experiments with nativescript and react native framework [online],” Diplomová práce, Masarykova univerzita, Fakulta informatiky, Brno, 2017 [cit. 2018-09-22]. [Online]. Available: [Dostupn\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{e\global\mathchardef\accent@spacefactor\spacefactor}\accent19e\egroup\spacefactor\accent@spacefactorzWWW\langlehttps://is.muni.cz/th/bkg22/\rangle](https://is.muni.cz/th/bkg22/)
- [62] Skia. (2018) Skia graphics library. [Online]. Available: <https://skia.org/>
- [63] P. Que, X. Guo, and M. Zhu, “A comprehensive comparison between hybrid and native app paradigms,” in *Computational Intelligence and Communication Networks (CICN), 2016 8th International Conference on*. IEEE, 2016, pp. 611–614.
- [64] R. Lublinerman and S. Tripakis, “Modularity vs. reusability: Code generation from synchronous block diagrams,” in *Proceedings of the conference on Design, automation and test in Europe*. ACM, 2008, pp. 1504–1509.
- [65] Facebook. (2018) React native build native mobile apps using javascript and react. [Online]. Available: <https://facebook.github.io/react-native/>

- [66] Dart. (2018) Dart packages. [Online]. Available: <https://pub.dartlang.org/>
- [67] Flutter. (2018) Using packages. [Online]. Available: <https://flutter.io/using-packages/>
- [68] I. Malavolta, S. Ruberto, T. Soru, and V. Terragni, “Hybrid mobile apps in the google play store: An exploratory investigation,” in *Proceedings of the Second ACM International Conference on Mobile Software Engineering and Systems*. IEEE Press, 2015, pp. 56–59.
- [69] W. Wu, “React native vs flutter, cross-platforms mobile application frameworks,” 2018.
- [70] Dart. (2018) Important concepts. [Online]. Available: <https://www.dartlang.org/guides/language/language-tour>
- [71] TIOBE. (2018) Tiobe index for september 2018. [Online]. Available: <https://www.tiobe.com/tiobe-index/>
- [72] B. Eisenman, *Learning React Native: Building Native Mobile Apps with JavaScript*. ” O’Reilly Media, Inc.”, 2015.
- [73] E. Shehab and H. Abdalla, “Manufacturing cost modelling for concurrent product development,” *Robotics and Computer-Integrated Manufacturing*, vol. 17, no. 4, pp. 341–353, 2001.
- [74] Cordova. (2015) Platform support. [Online]. Available: <https://cordova.apache.org/docs/en/latest/guide/support/index.html>
- [75] Adobe. (2016) Build amazing mobile apps powered by open web tech. [Online]. Available: <https://phonegap.com/>
- [76] Axway. (2017) Titanium mobile development environment. [Online]. Available: <https://www.appcelerator.com/Titanium/>
- [77] Google. (2018) How to use the play console. [Online]. Available: <https://support.google.com/googleplay/android-developer/answer/6112435>
- [78] Apple. (2018) Choosing a membership. [Online]. Available: <https://www.developer.apple.com/support/compare-memberships/>

- [79] A. Studio. (2018) Android studio. [Online]. Available: <https://developer.android.com/studio/>
- [80] Flutter. Build beautiful native apps in record time. [Online]. Available: <https://flutter.io/>
- [81] M. Bigio. (2016) Introducing hot reloading. [Online]. Available: <http://facebook.github.io/react-native/blog/2016/03/24/introducing-hot-reloading>
- [82] I. Pérez-Pérez, “Seguridad de aplicaciones híbridas para dispositivos móviles,” 2015.
- [83] V. N. Inukollu, D. D. Keshamoni, T. Kang, and M. Inukollu, “Factors influencing quality of mobile apps: Role of mobile app development life cycle,” *arXiv preprint arXiv:1410.4537*, 2014.
- [84] Oracle. (2017) Java(tm) ee 8 specification apis. [Online]. Available: <https://javaee.github.io/javaee-spec/javadocs/>
- [85] P. G. D. Group. (2019, November) PostgreSQL: The world’s most advanced open source relational database. [Online]. Available: <https://www.postgresql.org/>
- [86] P. development team. (2020) Php released. [Online]. Available: <https://www.php.net>
- [87] G. Developers. Lighthouse. [Online]. Available: <https://developers.google.com/web/tools/lighthouse>

Appendices

Appendix A

UML Diagram of the Flutter Recipe Cost Developed by the CG Created.

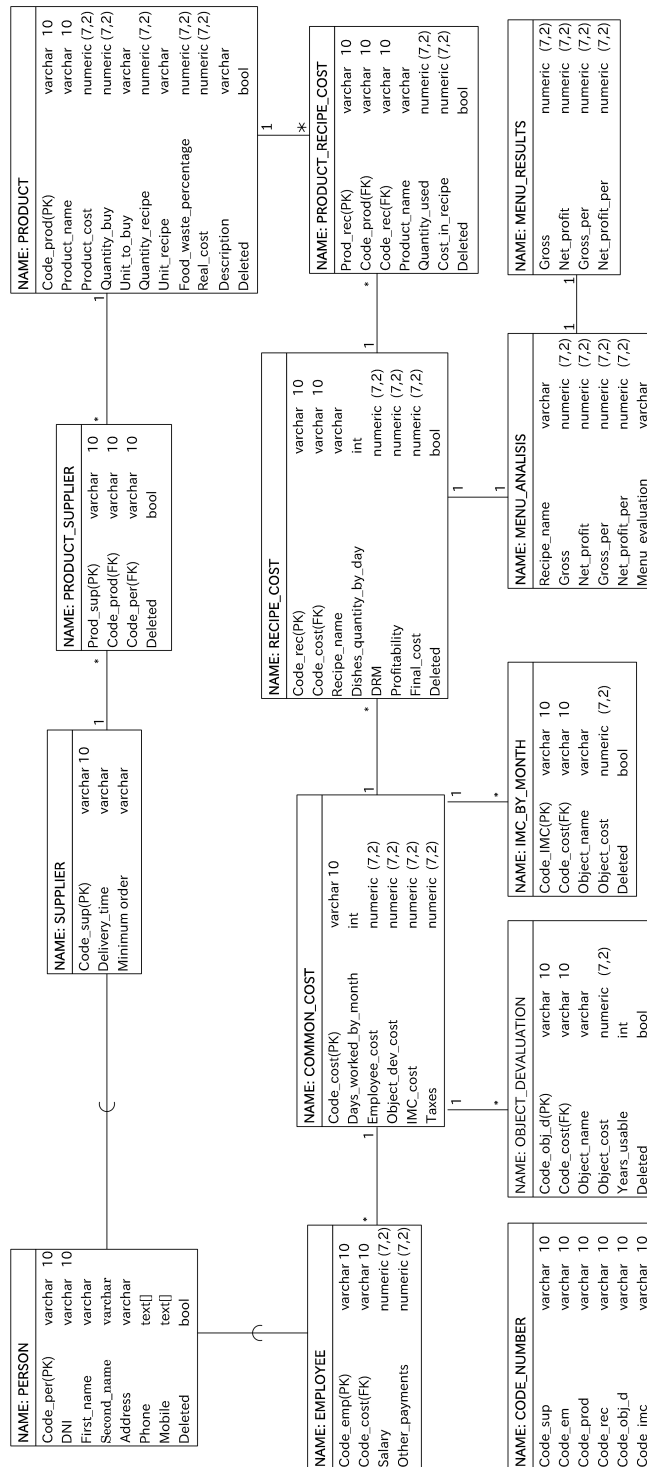


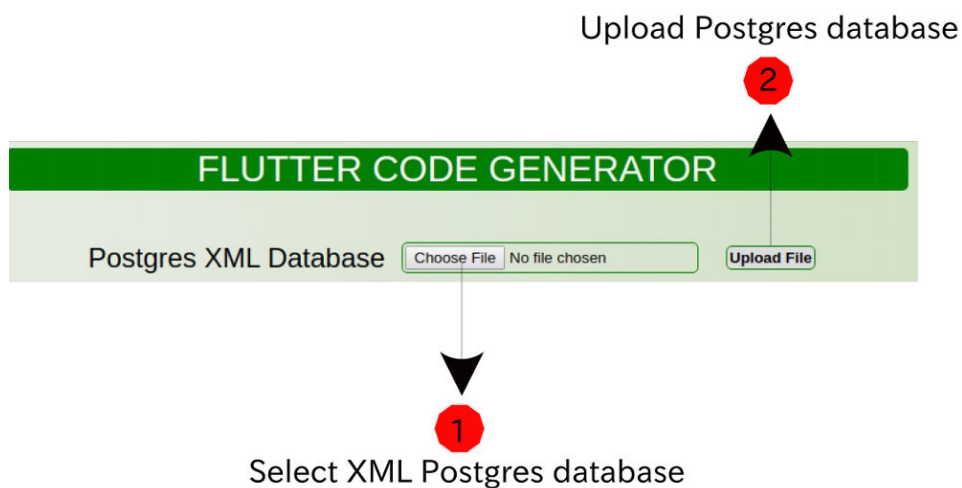
Figure A.1: UML diagram of the Flutter recipe cost application obtained by the CG.

Appendix B

User Manual

B.1 Update XML Postgres Database.


- 1) Button to select an XML Postgres Database.
- 2) Button to upload an XML Postgres Database.



B.2 Complete Application Information.

- 3) Button to select tables of the Postgres Database.
- 4) Information to connect with the Apache Server.

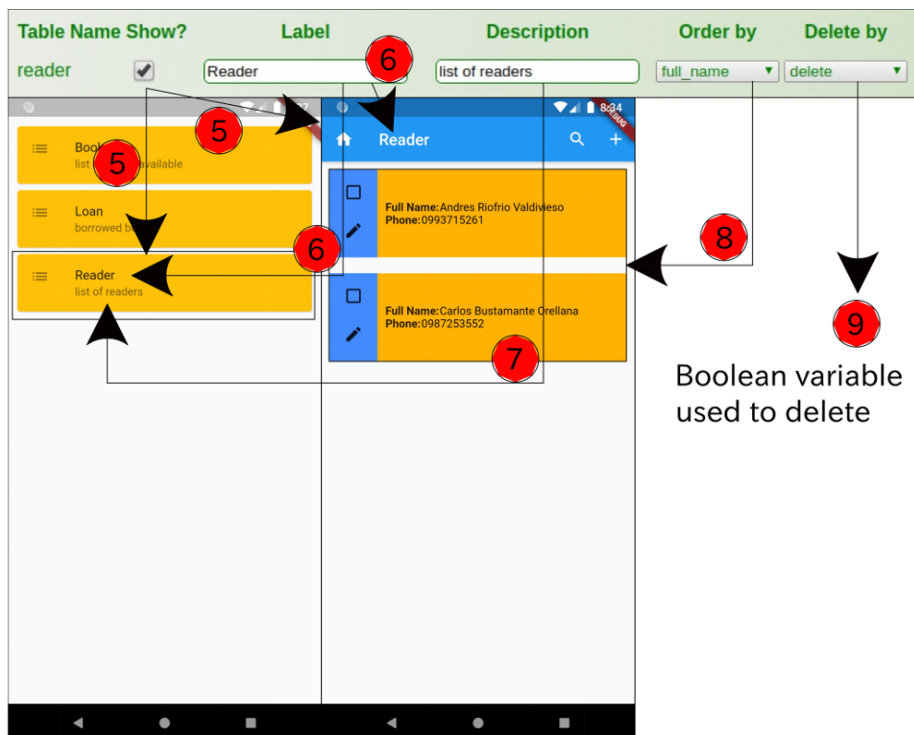
3 Select database tables



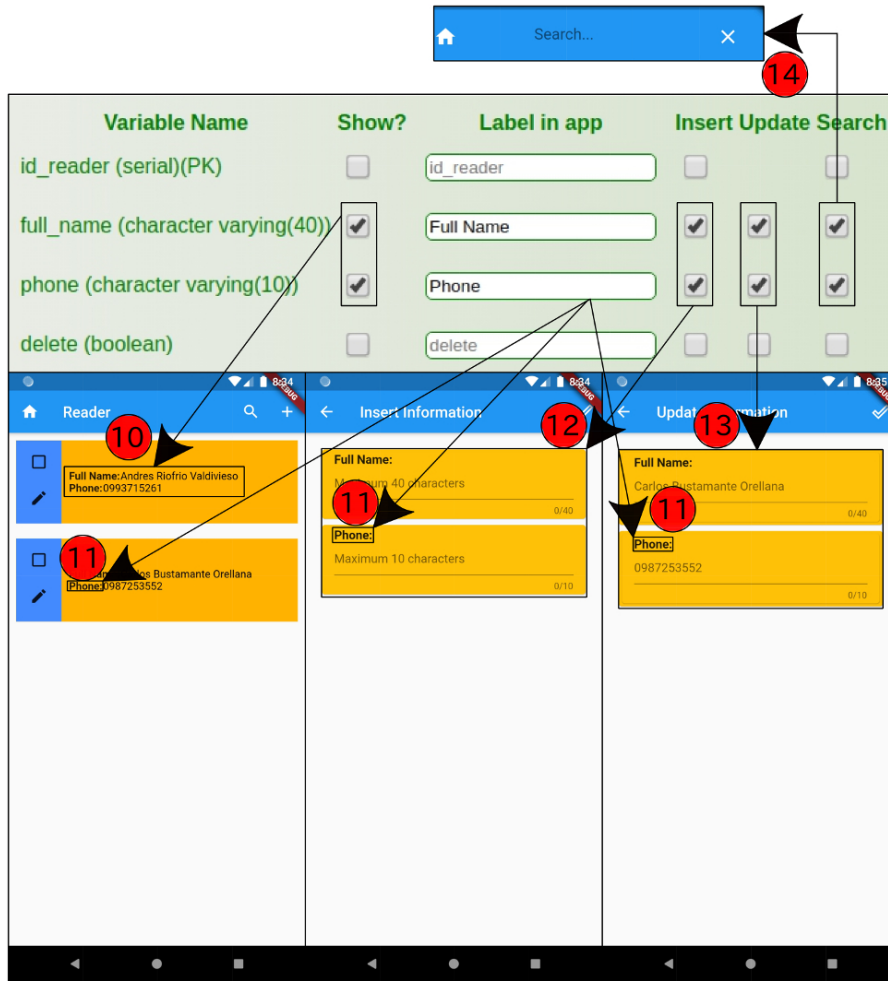
4

Information necessary to connect the server with the Postgres database

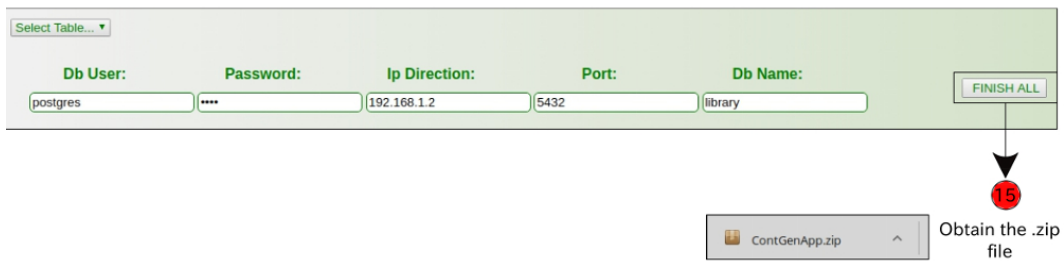
- 5) Create a button and window of the database table.
- 6) Title of the button and window of the before step.
- 7) Description of the button of the five-step.
- 8) Variable used to order the information of each window.
- 9) Boolean variable to delete information. This variable changes to true in order to do not show the information.



- 10) Selection of variables that will be showed.
- 11) The label of the variable for all windows where are used.
- 12) Variables used to insert data.
- 13) Variables used to update data.
- 14) Variables that will be used to search for information.

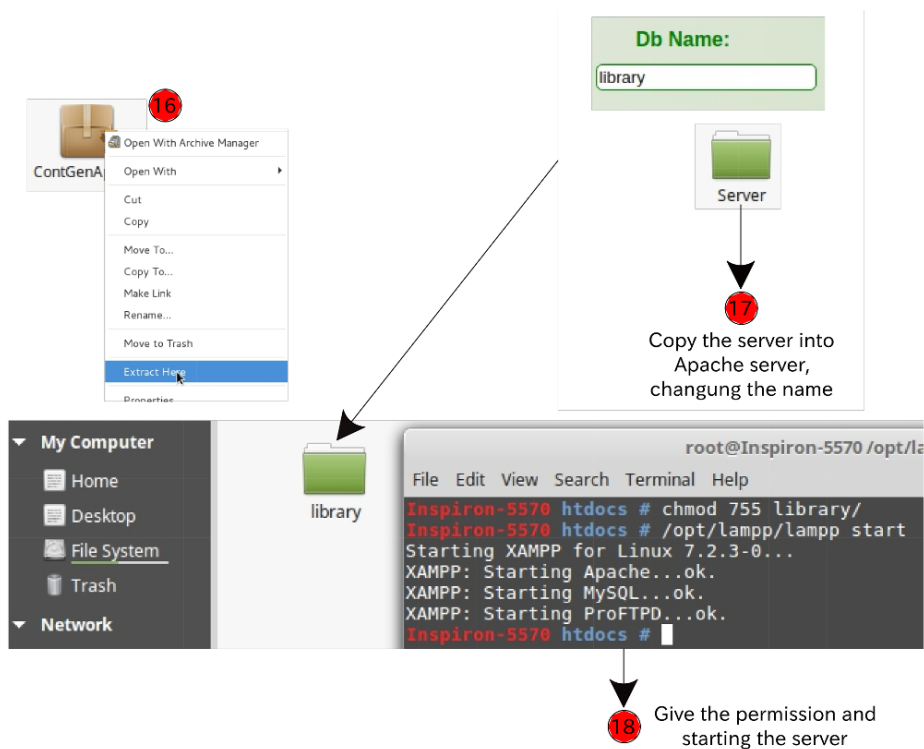


- 15) Finish the button to obtain the zip file.

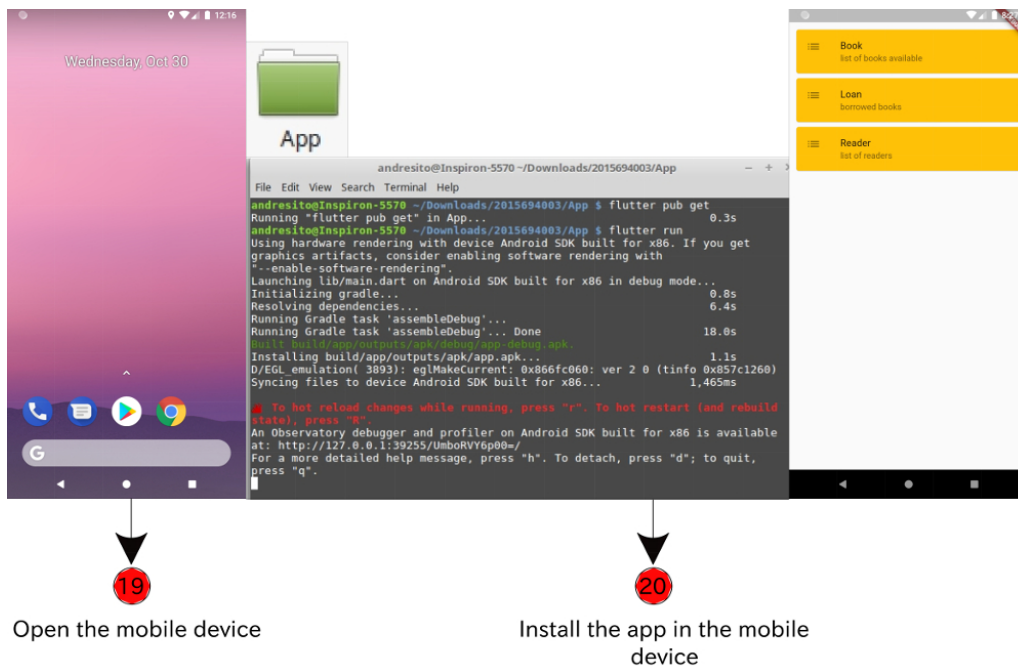


B.3 Run the Application

- 16) Extract the zip file.
- 17) Copy the server in the Apache. server that generally has the path: “/opt/lampp/htdocs”.
- 18) Permit to connect and start the server.



- 19) Open the virtual device or connect your mobile that needs to have the developer options.
- 20) Install the packages and run the application with the next Linux commands:
 - flutter pub get
 - flutter run



Appendix C

Technical Manual

C.1 Installation of the CG

The CG needs a server with NetBeans IDE 8.2 with Java EE 8 and Apache Tomcat 9.0.12. Furthermore, the “codeGenerator” file needs are imported and the “history” of NetBeans needs to be disabled because the process of generating a zip file could fill the history and consume all resources of the server. To run the CG is necessary to make some changes in the code that are showed in the Table C.1.

Package	Class	Line	Path Description
controller. Gen- Database. Post-gres	ContGenDatabase.java	40	Path to select where are stored the XML Postgres database
model.GenApp	ModGenApp.java	50, 229-232	Paths, where are stored templates, could be read the database and where will be created the zip files
model. Upda- teXML	ModUploadXML.java	43	The path where will be stored the XML database

Table C.1: CG Paths to update.

Now, is possible to run the CG and generates code taking account that the CG is connected with the user according to the sequence diagram showed in Fig. C.1

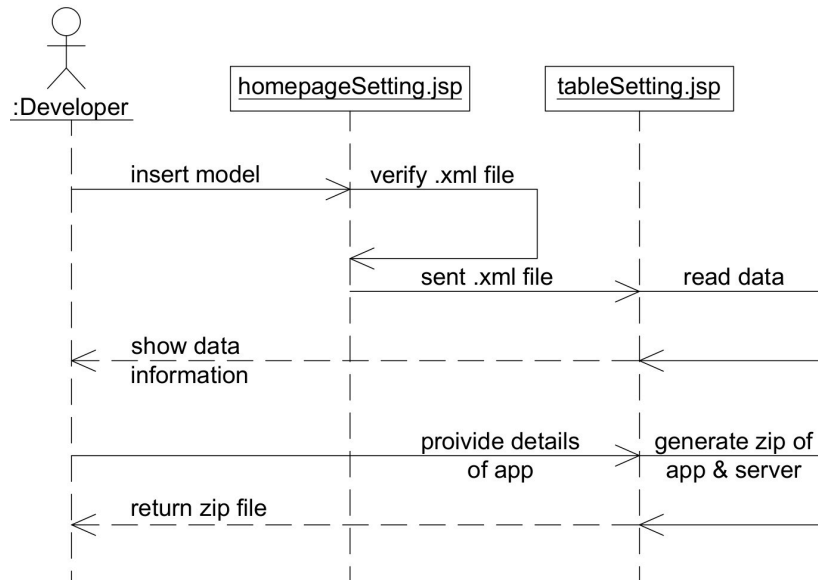


Figure C.1: Sequence diagram to create an application with the CG developed.

C.2 CG User Requirements

To develop a CG for mobiles was necessary first to analyze the following user requirements:

- **UCGR_1:** The CG shall allow to read a database model as is shown in Fig. C.2 to display the available tables with relations of one to one, one to many or many to many. Furthermore, it needs to accept serial, integer, numeric, boolean and varchar datatypes.

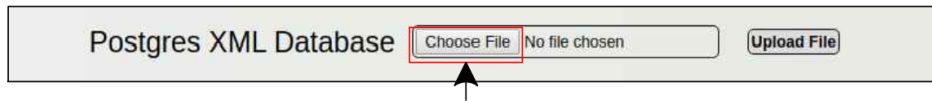


Figure C.2: CG user requirement number UCGR_1

- **UCGR_2:** The CG shall allow to select the database table that it shall be displayed in the mobile application as a select button. As is shown in the in Fig. C.3.

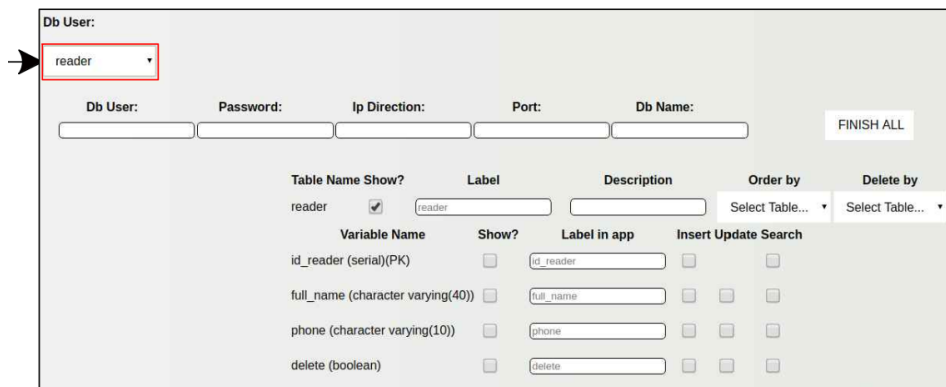


Figure C.3: CG user requirement number UCGR_2

- **UCGR_3:** Every table button and variables shall have as default the name from the database. The user could change this name and select what database tables and database variables shall be shown as Fig. C.4.

Table Name	Show?	Label	Description	Order by	Delete by
reader	<input checked="" type="checkbox"/>	reader		Select Table...	Select Table...

Variable Name	Show?	Label in app	Insert	Update	Search
id_reader (serial)(PK)	<input type="checkbox"/>	id_reader	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
full_name (character varying(40))	<input type="checkbox"/>	full_name	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
phone (character varying(10))	<input type="checkbox"/>	phone	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
delete (boolean)	<input type="checkbox"/>	delete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure C.4: CG user requirement number UCGR_3

- UCGR_4:** The CG shall allow to select the fields of each table to customize its display name and the insert, update and search operations. Except for primary and foreign keys that could not be updated. As is shown in the in Fig. C.5.

Table Name	Show?	Label	Description	Order by	Delete by
reader	<input checked="" type="checkbox"/>	reader		Select Table...	Select Table...

Variable Name	Show?	Label in app	Insert	Update	Search
id_reader (serial)(PK)	<input type="checkbox"/>	id_reader	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
full_name (character varying(40))	<input type="checkbox"/>	full_name	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
phone (character varying(10))	<input type="checkbox"/>	phone	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
delete (boolean)	<input type="checkbox"/>	delete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure C.5: CG user requirement number UCGR_4

- UCGR_5:** The information showed to each database table shall be ordered from its variables. Then, a select list shall be showed to select the variable used to order the information. As is shown in the in Fig. C.6.

Table Name Show?	Label	Description	Order by	Delete by
reader <input checked="" type="checkbox"/>	reader		Select Table... ▼	Select Table... ▼

Variable Name	Show?	Label in app	Insert	Update	Search
id_reader (serial)(PK)	<input type="checkbox"/>	id_reader	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
full_name (character varying(40))	<input type="checkbox"/>	full_name	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
phone (character varying(10))	<input type="checkbox"/>	phone	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
delete (boolean)	<input type="checkbox"/>	delete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure C.6: CG user requirement number UCGR_5

- **UCGR_6:** The delete option shall be a boolean variable that could be changed to true to do not show the information of this tuple. Then, a list of boolean variables for each table shall be showed to the user. As is shown in the in Fig. C.7.

Figure C.7: CG user requirement number UCGR_6

- **UCGR_7:** The CG shall allow insert database information as username, password, ip direction, port and database name. This information shall be used to create the server and the connection with its mobile application. As is shown in the in Fig. C.8.

Figure C.8: CG user requirement number UCGR_7

C.3 Templates definition for Mobile Application Interface and Code

This stage contemplates the definition of the mobile application user interface that will be created by the CG. Therefore, a standard user interface is created to build the templates based on the definitions summarized in the Fig.C.9, Fig. C.10, and the Fig. C.11:

- **DEF_1:** The main window of the application will allow buttons with titles and subtitles. The subtitles will be optional to insert.
- **DEF_2:** Each table of the user database could have one window in the application that will be opened from one button of the main window.
- **DEF_3:** The windows created in the application through the database will have a search and checkbox to delete information. Furthermore, this window will have one button to delete, one button to update, and one button to return at the main window of the application.

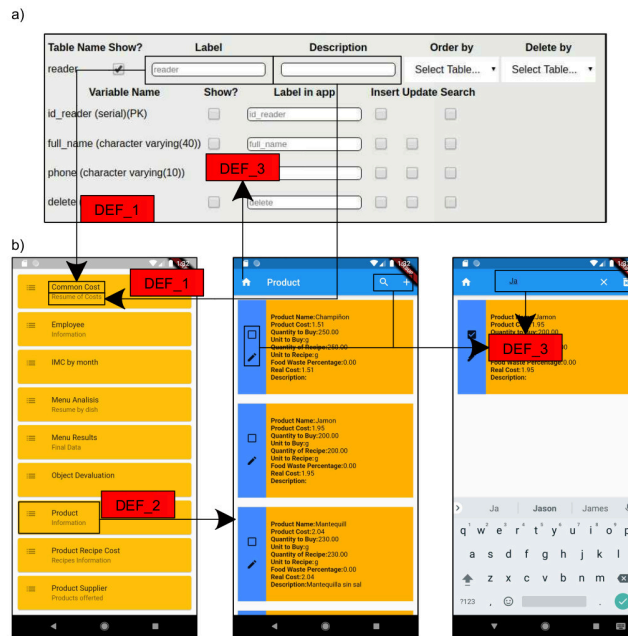


Figure C.9: Design of a Standard UI based on definition DEF_1-DEF_3. a) tablePageSetting.jsp and b) mobile application.

- **DEF_4:** The delete and update button will be the same in the application to all windows created through the tables of the database. Then, this will be complete dynamically.



Figure C.10: Definition DEF_4.

- **DEF_5:** The foreign keys attributes of the application will be buttons with the information of their primary keys variables.
- **DEF_6:** Users will not be confused with “primary keys ids” in the application. Then, foreign key buttons will be shown information related to the primary key button.
- **DEF_7:** The update, delete, and insert button will have a dialog box to confirm the user requirement.

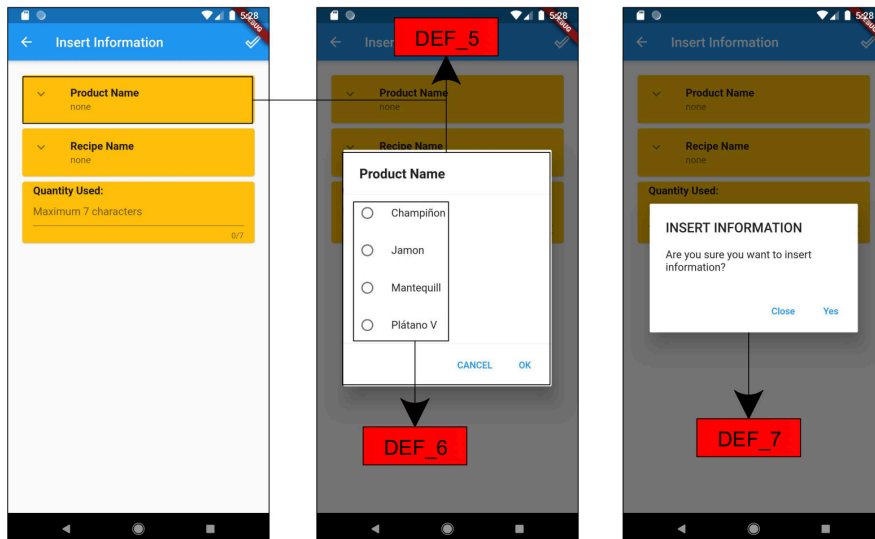


Figure C.11: Definition DEF_5 until DEF_7.

After defining the standard mobile user interface that will produce the CG, the templates for mobile applications code are extracted and used posteriorly in the implementation. The necessary templates contain tags bounded by “<< >>”. For instance, << *dbName* >> will be replaced with the name of the database inserted by the user when the code is generated. Then, PHP and Flutter templates were created and posteriorly used through writeTemplate Java EE method of the class genTemplate that receives the path of the template, the path of the user code, the words that will be changed, and the new words to create a new file.

- **PHP templates:** These templates allow connecting the Apache server with the database and mobile application. For instance, the code of the dbInformation.php shown in the Algorithm C.1 is a template that contains the information necessary to connect the database with the server as ip, port, database name, user name, and password.

```

1 <?PHP
2 $conexion = pg_connect("host=<<ip>> port=<<port>> dbname=<<dbName>> user=<<dbUser>>
   password=<<password>>")
3 or die('Lost Connection: ' . pg_last_error ());
4 ?>

```

Algorithm C.1: PHP template to connect database with apache server.

This template will be replaced by the information of the userDb object of the Algorithm C.2 through writeTemplate Java EE method.

```

1 // Creation dbInformation.php
2 ArrayList<String> wordsToChange = new ArrayList();
3 wordsToChange.add("<<dbUser>>");
4 wordsToChange.add("<<password>>");
5 wordsToChange.add("<<ip>>");
6 wordsToChange.add("<<port>>");
7 wordsToChange.add("<<dbName>>");
8 ArrayList<String> newWords = new ArrayList();
9 newWords.add(userDb.getDbUser());
10 newWords.add(userDb.getPassword());
11 newWords.add(userDb.getIpDirection ());
12 newWords.add(userDb.getPort());
13 newWords.add(userDb.getName());
14 genTemplate.writeTemplate (pathToWrite, pathToCopy, wordsToChange, newWords);

```

Algorithm C.2: Java code to replace PHP template of the Algorithm C.1.

- **Flutter template:** It is a template of Flutter project that has tags to create mobile application according to user requirements. For instance, the tags `<< appDescription >>`, `<< appNames >>`, and `<< listOfScreen >>` are used to create the buttons of the mobile application main window. This tags are shown in the Algorithm C.3.

```

1 //Creation list of screens
2 class _Routes extends State<DynScreens> {
3   //Global variables
4   List appDescription = [<<appDescription>>];
5   List appScreenNames = [<<appNames>>];
6   List listOfScreen = [<<listOfScreen>>];
7   @override
8   //View of principal screen
9   Widget build(BuildContext context) {
10    return Scaffold(
11      //List of widges
12      body: Center(
13        child: Container(
14          padding: EdgeInsets.all(8.0),
15          child: ListView.builder(
16            itemCount: appScreenNames.length,
17            itemBuilder: (BuildContext ctxt, int i) {
18              return Column(
19                children: <Widget>[
20                  //Button to each screen
21                  Card(
22                    color: Colors.amber,
23                    child: ListTile(
24                      title : Text(appScreenNames[i]),
25                      subtitle : Text(appDescription[i]),
26                      leading: const Icon(Icons.list),
27                      onTap: () {
28                        Navigator.push(
29                          context,
30                          MaterialPageRoute(builder : (context) => listOfScreen[i]),
31                        );
32                      },
33                    ),
34                  ),
35                ],
36              );
37            })),
38    ),

```

```

39     ),
40     );
41   }
42 }

```

Algorithm C.3: Flutter template to create buttons of the main window of the mobile application.

This template will be replaced with user information inserted in the CG through the use of the writeTemplate Java EE method as shown Algorithm C.4.

```

1 // Creation of main window buttons
2 ArrayList<String> wordsToChange = new ArrayList();
3 wordsToChange.add("<<appDescription>>");
4 wordsToChange.add("<<appNames>>");
5 wordsToChange.add("<<listOfScreen>>");
6 ArrayList<String> newWords = new ArrayList();
7 newWords.add(userDb.getAppDescription());
8 newWords.add(userDb.getAppNames());
9 newWords.add(userDb.getListOfScreen());
10 genTemplate.writeTemplate(pathToWrite, pathToCopy, wordsToChange, newWords);

```

Algorithm C.4: Java code to replace Flutter template of the Algorithm C.3.

These templates need a copy inside a folder that shall be given to the user. This process is carried after the user makes click on the finish button. Then, to generate the zip file, some java packages were used inside JavaEE as “java.io” to read templates, “java.lang.Runtime” to execute Linux commands as copy templates, and “java.util.zip” to generate the zip file.

C.4 Architectural Pattern

The CG has a MVC architectural pattern that connects the model and the view through the controller. In the view is possible to update the XML Postgres database and complete all the information for generating the application. Meanwhile, the model creates a zip that contains the application and the server.

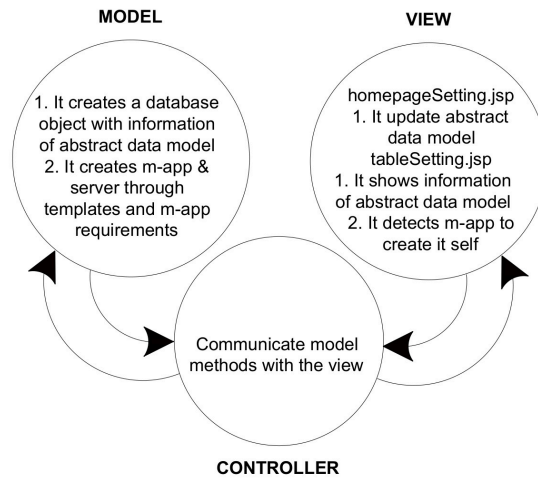


Figure C.12: MVC architecture of the CG.

This architecture is distributed in packages showed in Fig. C.13.

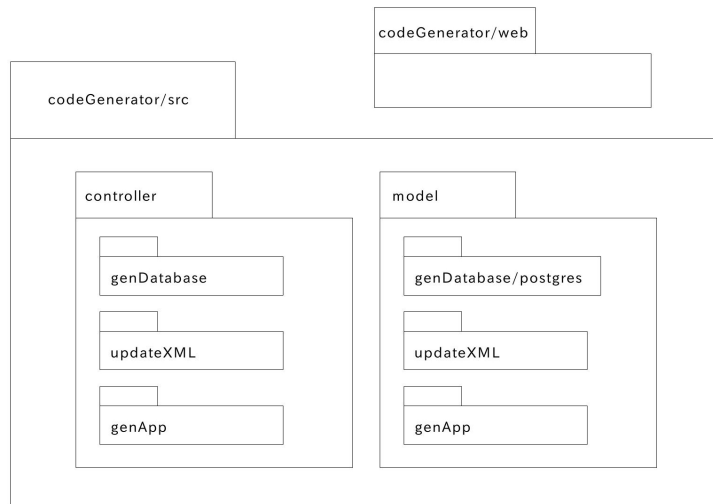


Figure C.13: Package diagram of the CG.

The java classes are interconnected as Fig. C.14 to create the zip file with the application and server.

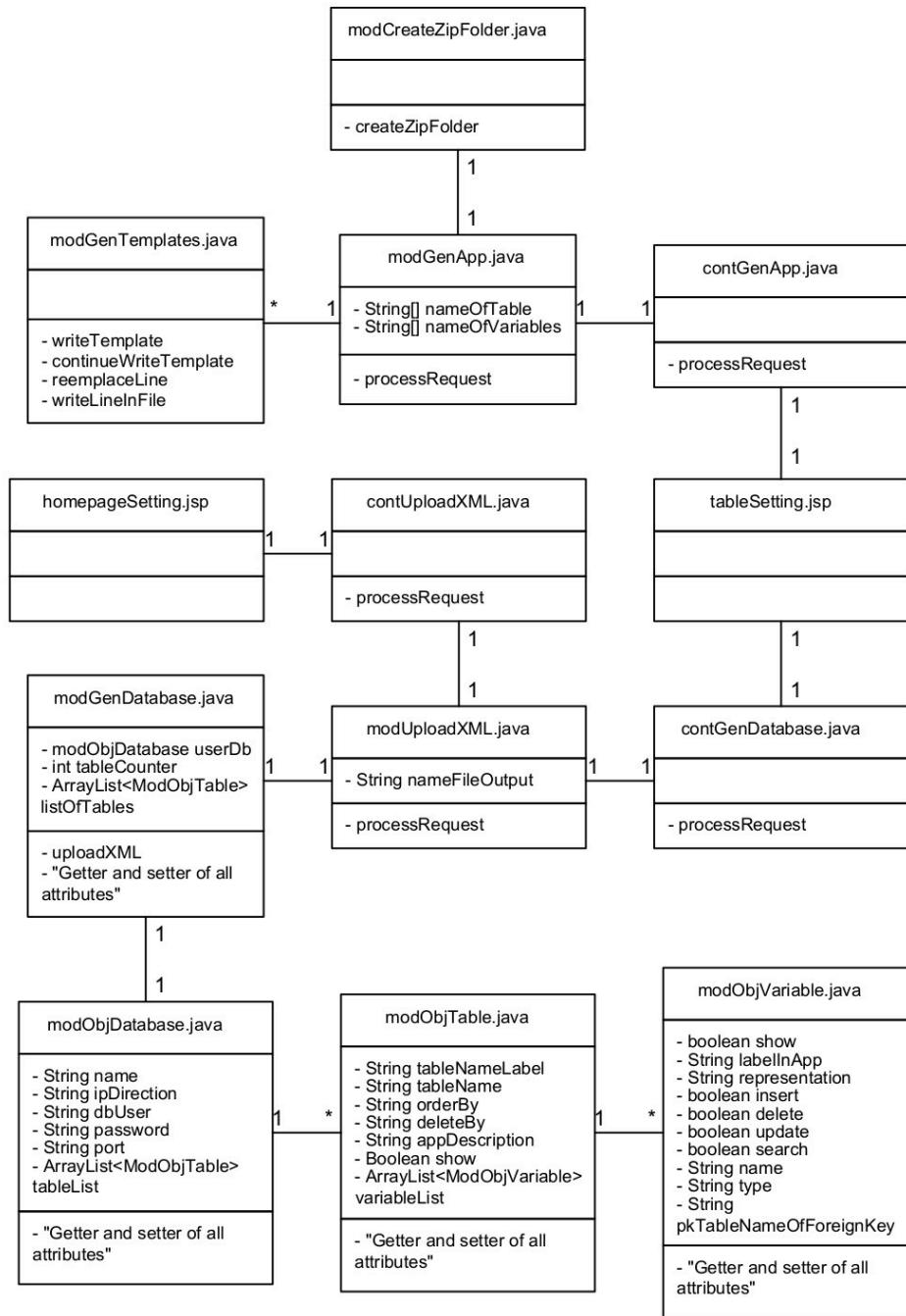


Figure C.14: Class diagram of the CG.

Furthermore, during the creation of the back-end of the CG was necessary use templates for

the mobile applications according to user specifications with a MVC architectural pattern that is shown in Fig. C.15.

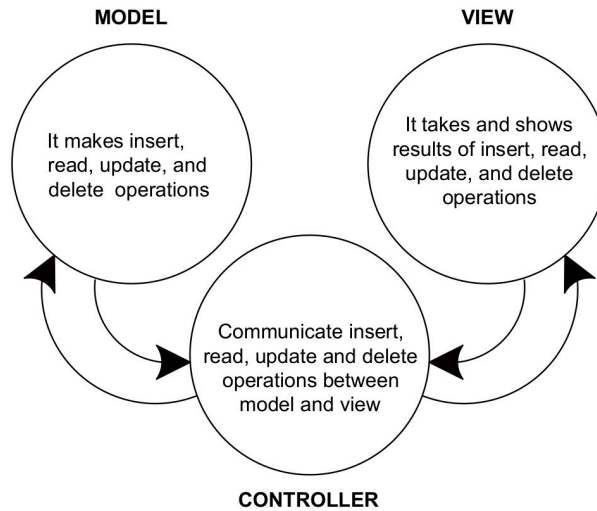


Figure C.15: MVC for mobile applications developed with the CG

This MVC architectural pattern is implemented inside the lib package of the mobile application template that is shown in the package diagram of the Fig. C.16 where the main.dart is the main window that contains all buttons selected in the CG. The insert.dart and update.dart files are dynamic windows used by the screen.dart that changes its name according to user requirements and it is copy according to the number of windows that select the user to shown according to database tables.

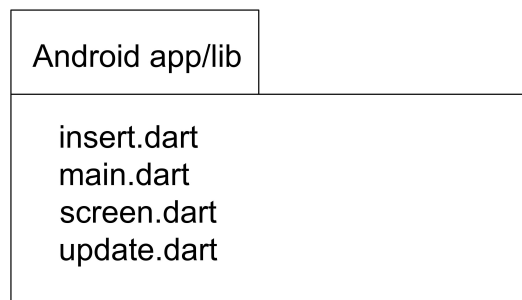


Figure C.16: Package diagram of the mobile applications

Moreover, the class diagram used by the mobile applications is showed in the Fig. C.17

taking account that the user could create n windows that has the same connection using one insert.dart, update.dart and main.dart files.

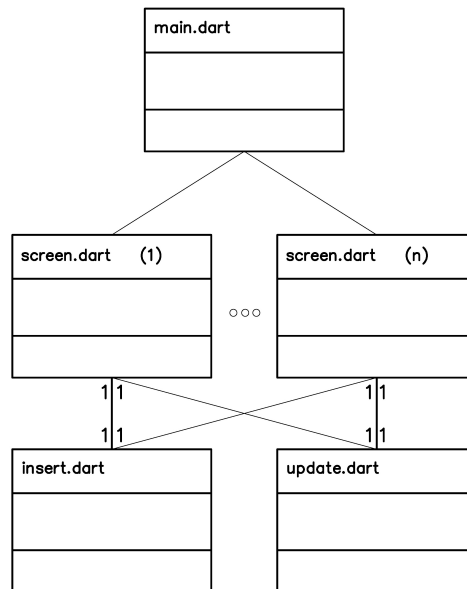


Figure C.17: Class diagram of the mobile applications

Appendix D

Thesis Web Site

The whole thesis (code generator, code generated, templates, video, etc.) is uploaded into a web site for free access. This web site is the property of my adviser Ph.D. Lorena de los Angeles Guachi. The web site has been developed in Google Sites, and the link for this research could be obtained from:

<https://sites.google.com/site/degreethesislorenaguachi/2020-andr%C3%A9s-riofr%C3%A9do-code-generator-for-mobile-apps>