PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

ESCUELA DE SISTEMAS

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS Y COMPUTACIÓN

"DESARROLLO DE UN APLICATIVO MÓVIL PARA LA GESTIÓN DE CUADERNOS DE CAMPO. CASO DE ESTUDIO MUSEO QCAZ DE LA PUCE"

AUTOR:

JUAN ESTEBAN CANELOS FLORES

DIRECTOR:

MSC. DAMIÁN ANIBAL NICOLALDE RODRÍGUEZ

QUITO, 2017

DEDICATORIA

Quisiera dedicar este trabajo a mis papás Ana María y Ramiro, quienes me han apoyado incondicionalmente durante toda mi carrera, y me han esperado pacientemente durante todo este proceso tan importante en mi vida.

AGRADECIMIENTOS

Quisiera dedicar este trabajo a mi mamá Ana María y a mi papá Ramiro, quienes me han brindado su apoyo incondicional a lo largo de toda mi carrera y toda mi vida. Sé que siempre podré contar con ustedes y que siempre confiarán en mí.

A mi hermano Daniel, con quien he compartido muchos momentos, en especial durante el final de mi carrera, y quien se ha convertido en mi mejor amigo y compañero incondicional.

A mis amigos de siempre, que a pesar de todo hicieron de mi estadía en la universidad la mejor época de mi vida.

A aquellos grandes profesores que aumentaron mis conocimientos y dieron vida a mi amor por esta carrera.

A todas aquellas personas especiales en mi vida, y aquellas que la vida ha ido cruzando recientemente en mi camino.

RESUMEN

Este trabajo se ha centrado en ayudar a optimizar el proceso de manejo de cuadernos de campo, llevado a cabo por los alumnos, docente e investigadores de biología del Museo QCAZ de la Pontifica Universidad Católica del Ecuador, buscando aprovechar las tecnologías de los smartphones.

En el capítulo 1 se presenta la fundamentación teórica sobre los conceptos, términos y herramientas utilizados durante esta disertación. En el capítulo 2 se exponen los antecedentes y se recogen los requerimientos levantados para el desarrollo de la aplicación. En el capítulo 3 se recolecta toda la documentación realizada durante el desarrollo del aplicativo, bajo la metodología de desarrollo ágil utilizada. Finalmente, en el capítulo 4 se presenta toda la documentación necesaria para la implementación de la aplicación.

Índice

1. Fundamentación Teórica 12
   1.1. Software 12
      1.1.1. Aplicación Móvil 12
   1.2. Ingeniería de Software 13
      1.2.1. Ciclo de Vida del Software 14
      1.2.2. Especificación de Requerimientos de Software 15
   1.3. Bases de Datos 16
      1.3.1. Sistemas Gestores de Bases de Datos (SGBD) 16
      1.3.2. Base de Datos Relacional 17
      1.3.3. SQLite 17
   1.4. Lenguajes de Programación 18
      1.4.1. Java 19
      1.4.2. C# 21
   1.5. .NET Framework 22
      1.5.1. ADO.NET 23
      1.5.2. LINQ 23
      1.5.3. Entity Framework 23
   1.6. IDE 24
      1.6.1. Android Studio 25
      1.6.2. Visual Studio 26
   1.7. Servicios Web 27
      1.7.1. SOAP 28
      1.7.2. REST 29
      1.7.3. SOAP vs REST 29
   1.8. Metodologías de Desarrollo 30
      1.8.1. Extreme Programming 30
2. Requerimientos 34
   2.1. Diagnóstico de la Situación Actual 34

2.2. Levantamiento de Requerimientos 35
2.3. Sistema Propuesto 36
2.3.1. Casos de Uso 37
2.3.1.1. Caso de Uso General 37
2.3.1.2. Caso de Uso Específico 37
2.4. Entorno del Software 38
3. Desarrollo 39
3.1. Primera iteración 39
3.1.1. Análisis 39
3.1.2. Diseño 39
3.1.3. Codificación 41
3.2. Segunda iteración 42
3.2.1. Análisis 42
3.2.2. Diseño 42
3.2.3. Codificación 44
3.3. Tercera iteración 45
3.3.1. Análisis 45
3.3.2. Diseño 46
3.3.3. Codificación 48
3.4. Cuarta iteración 49
3.4.1. Análisis 49
3.4.2. Diseño 51
3.4.3. Codificación 53
3.5. Quinta iteración 54
3.5.1. Análisis 54
3.5.2. Diseño 55
3.5.3. Codificación 57
4. Implementación 58
4.1. Diagrama de Despliegue 58

4.2. Instalación y Configuración 58
4.3. Manual de Usuario 59

Índice de Ilustraciones

1. Fundamentación Teórica

1.1. Cuadro del crecimiento en descargas de las apps a nivel mundial 13

1.2. Ciclo de vida del software 15

1.3. Arquitectura cliente/servidor convencional 18

1.4. Arquitectura sin servidor de SQLite 18

1.5. Índice de popularidad de lenguajes de programación para julio 2017 20

1.6. Índice TIOBE de popularidad de lenguajes de programación para julio 2017 21

1.7. Componentes de .NET Framework 22

1.8. Escenarios de uso de .NET Framework 24

1.9. Ejemplo de servicios web en funcionamiento 27

1.10. Desarrollo de un proyecto con Extreme Programming 31

1.11. Iteración en un proyecto con Extreme Programming 32

2. Requerimientos

2.1. Caso de uso general 37

2.2. Caso de uso específico 37

2.3. Distribución de versiones de plataformas/API de Android en el mercado 38

3. Desarrollo

3.1. Diagrama de clases en primera iteración 39

3.2. Diagrama conceptual en primera iteración 40

3.3. Diagrama conceptual del aplicativo móvil en primera iteración 40

3.4. Prototipo de interfaz gráfica de pantalla de ingreso 40

3.5. Diagrama de clases en segunda iteración 42

3.6. Diagrama conceptual en segunda iteración 43

3.7. Diagrama conceptual del aplicativo móvil en segunda iteración 43

3.8. Prototipo de interfaz gráfica de pantalla de ingreso de cuaderno de campo 44

3.9. Prototipo de interfaz gráfica de pantalla de ingreso de cuaderno de campo 44
3.10. Diagrama de clases en tercera iteración 46
3.11. Diagrama conceptual en tercera iteración 46
3.12. Diagrama conceptual del aplicativo móvil en tercera iteración 47
3.13. Prototipo de interfaz gráfica de pantalla de uso de sensores 47
3.14. Diagrama de clases en cuarta iteración 51
3.15. Diagrama conceptual en cuarta iteración 51
3.16. Diagrama conceptual del aplicativo móvil en cuarta iteración 52
3.17. Prototipo de interfaz gráfica de pantalla de búsqueda de cuadernos de campo 52
3.18. Diagrama de clases en quinta iteración 55
3.19. Diagrama conceptual en quinta iteración 55
3.20. Diagrama conceptual del aplicativo móvil en quinta iteración 56
3.21. Prototipo de interfaz gráfica de pantalla de modificación y eliminación de cuadernos de campo 56
4. Implementación
4.1. Diagrama de despliegue 58

Índice de Tablas

2. Requerimientos

2.1. Historia de usuario US1 35

2.2. Historia de usuario US2 35

2.3. Historia de usuario US3 35

2.4. Historia de usuario US4 36

2.5. Historia de usuario US5 36

3. Desarrollo

3.1. Criterios de aceptación de historia de usuario US1 39

3.2. Pruebas de historia de usuario US1 41

3.3. Criterios de aceptación de historia de usuario US2 42

3.4. Pruebas de historia de usuario US2 44

3.5. Criterios de aceptación de historia de usuario US3 45

3.6. Pruebas de historia de usuario US3 48

3.7. Criterios de aceptación de historia de usuario US3 49

3.8. Criterios de aceptación de historia de usuario US4 50

3.9. Pruebas de historia de usuario US3 53

3.10. Pruebas de historia de usuario US4 53

3.11. Criterios de aceptación de historia de usuario US5 54

3.12. Pruebas de historia de usuario US5 57

1. Fundamentación Teórica

1.1. Software

Es un producto que consiste en el conjunto de instrucciones, estructuras de datos y, generalmente, su documentación asociada, que "construyen los programadores profesionales y al que después le dan mantenimiento durante un largo tiempo. Incluye programas que se ejecutan en una computadora de cualquier tamaño y arquitectura, contenido que se presenta a medida que se ejecutan los programas de cómputo e información descriptiva tanto en una copia dura como en formatos virtuales que engloban virtualmente a cualesquiera medios electrónicos." (Pressman, 2010)

Existen diversos tipos de software según su funcionalidad y su complejidad, como por ejemplo softwares de sistema, softwares de gestión, softwares científicos, softwares empotrados, sistemas operativos, entre otros.

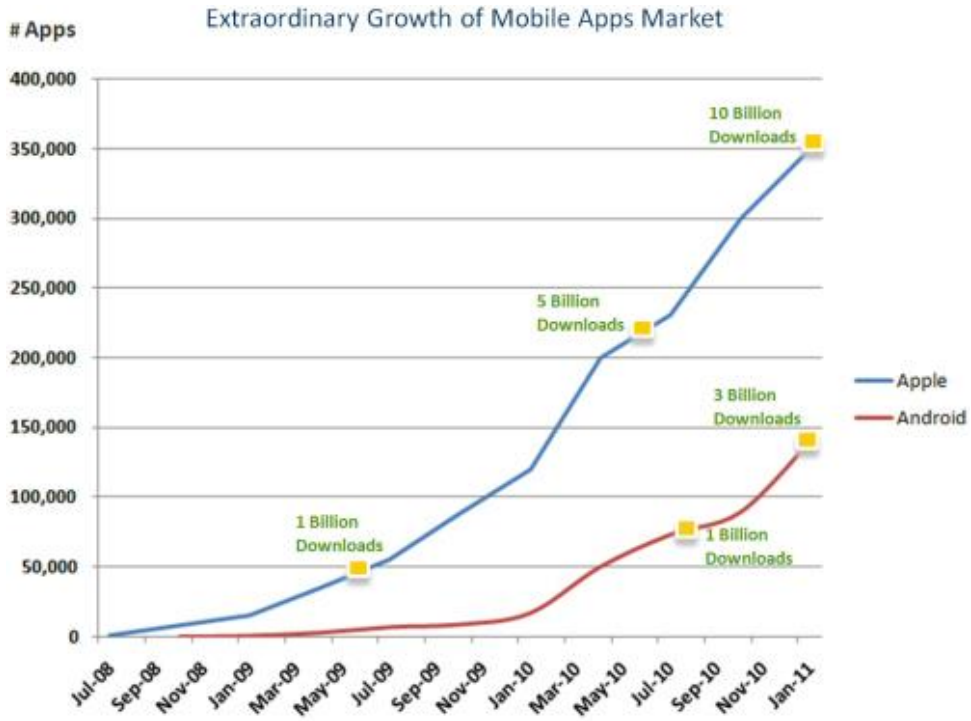
1.1.1. Aplicación Móvil

Las aplicaciones móviles, aplicativos móviles, o apps por su nombre en inglés, son aplicaciones informáticas diseñadas para instalarse y ejecutarse en dispositivos móviles, como los smartphones y las tabletas. Su funcionalidad puede ser de cualquier tipo: profesional, educativa, informativa, administradora, entretenimiento y ocio, comercial, comunicación, entre muchas otras.

Las aplicaciones móviles empezaron a surgir a mediados de los años 90 como programas para celulares, cuyas funciones eran básicas y sus diseños eran sencillos. Estas primeras aplicaciones cumplían tareas como el manejo de ringtones, videojuegos (como el famoso juego Snake), calendarios y agendas personales.

La evolución de las apps vio un incremento significativo en el año 2000, con la irrupción de las tecnologías WAP (protocolo de aplicaciones inalámbricas) y EDGE (tasas de datos mejoradas para la evolución del GSM), que permitieron una mayor capacidad para la descarga de videojuegos para celulares. Sin embargo, no fue hasta el año 2008, cuando aparecieron las plataformas de distribución digitales Android Market (predecesora de Google Play o Play Store, para dispositivos con sistema operativo Android) y App Store (para dispositivos con sistema operativo iOS), que las aplicaciones móviles alcanzaron su verdadero auge.

Ilustración 1.1 – Cuadro del crecimiento en descargas de las apps a nivel mundial



McGratha, A. (2011, febrero). *Extraordinary Growth of Mobile Apps Market.* Recuperado de https://mcgratha.wordpress.com/2011/02/18/there-is-an-ecm-app-for-that/

Hoy en día las aplicaciones forman parte esencial, e incluso indispensable de la vida cotidiana de mucha gente y empresas. La industria de las apps ha crecido al punto que en el año 2016 sus ingresos llegaron a los 51 billones de dólares, superando a la industria de la música que obtuvo ingresos por 46 millones de dólares. Para el año 2020, se espera que las descargas de aplicaciones lleguen a 284 billones, generando ingresos por 100 billones de dólares. (McGoogan , 2016)

1.2. Ingeniería de Software

Disciplina que estudia el conjunto de métodos, herramientas y técnicas involucradas en el desarrollo completo de software. La IEEE define a la ingeniería de software como "la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software. Es decir, la aplicación de ingeniería al software." (IEEE, 1990) El objetivo final es que el software resultante sea de calidad, cumpla con las necesidades del cliente o del usuario, sea confiable y pueda proveer el rendimiento adecuado.

"La ingeniería de software está formada por un proceso, un conjunto de métodos (prácticas) y un arreglo de herramientas que permite a los profesionales elaborar software de cómputo de alta calidad." (Pressman, 2010)

La ingeniería de software abarca todos los aspectos de la producción del software, incluyendo el análisis previo de los requisitos, el diseño, el desarrollo, las pruebas, la implementación y el soporte posterior a finalización del producto. Debido a la gran cantidad de elementos que involucra, Scott Whitmire sostiene que la ingeniería de software, más que ser una disciplina o un cuerpo de conocimiento, es un verbo, una palabra de acción, una manera de abordar un problema. (Whitmire, s.f.)

El término ingeniería de software fue usado por primera vez por Fritz Bauer, en octubre de 1968 durante el comité de Ciencia de la OTAN, celebrado en Alemania. Los problemas principales del software que hicieron necesario un cambio en el enfoque aplicado y llevaron a la aparición de la ingeniería de software son:

- Largos tiempos de desarrollos
- Costos elevados de desarrollo
- Imposibilidad de encontrar todos los errores en el software antes de entregar al cliente
- Dificultad de constatar el progreso conforme se desarrolla el software

1.2.1. Ciclo de Vida del Software

Según la IEEE, en su estándar para los procesos de ciclo de vida del software IEEE 1074, el ciclo de vida es “una aproximación a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software.” (IEEE, 1997)

Según la ISO, en su estándar para los procesos de ciclo de vida del software ISO/IEC 12207, es “un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso.” (ISO, 2008)

Ilustración 1.2 – Ciclo de vida del software

RRK Global Engineering. (2016). *Ciclo de Vida del Software.* Recuperado de http://www.rrkge.com/cvsw.html

1.2.2. Especificación de Requerimientos de Software

La especificación de requerimientos de software, o SRS (por sus siglas en inglés, Software Requirements Specification), es un documento que permite a los ingenieros de diseño comunicarse con el cliente, describiendo de manera clara y no ambigua el alcance y el funcionamiento del proyecto de software a realizarse, documentado los acuerdos entre el cliente y el grupo de desarrollo. El SRS debe plantear claramente lo que el sistema hará y no hará, de manera que se asegure el cumplimiento de las exigencias estipuladas y se controle las insistencias de nuevas funcionalidades por parte del cliente.

Un documento de SRS óptimo provee los siguientes beneficios, tanto a los proveedores, como a los clientes, y cualquier otro individuo involucrado (IEEE, 1998):

- Define las funcionalidades del software, estableciendo una base contractual entre el cliente y el proveedor que les permitirá definir si el software final cumple con las necesidades especificadas del cliente.
- Al ser un documento preparado por las partes involucradas en el proyecto, los requisitos se definen claramente desde un inicio, reduciendo el esfuerzo de desarrollo y minimizando la posibilidad de futuros rediseños, recodificaciones y nuevas pruebas.
- Provee una base realista para una mejor estimación de tiempos y costos del proyecto.

- Sirve como una línea base para la validación y verificación del producto final.
- Facilita el paso del software a nuevos usuarios, tanto a los clientes (usuarios dentro de sus mismas organizaciones) como a los proveedores (nuevos clientes).
- Sirve como base para futuras mejoras del software final.

1.3. Bases de Datos

El almacenamiento de la información siempre ha sido una tarea esencial para el hombre, por lo cual siempre se han ido desarrollando técnicas y herramientas para este fin. Con la aparición de los primeros sistemas informáticos, la información empezó a ser almacenada en cintas perforadas y en archivos. Eventualmente, la creciente cantidad de información a ser almacenada, accedida y procesada hizo necesario integrarla en un solo lugar. Fue a mediados de la década de 1960 cuando aparecieron los primeros sistemas de bases de datos.

Una base de datos es una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular. Los principales objetivos de las bases de datos son crear independencia de los datos, reducir su redundancia, mantener su integridad y consistencia, permitir el acceso concurrente, y brindar seguridad y respaldos. (Pérez Valdés, 2007)

Una base de datos también puede verse como "un conjunto de datos dispuestos con el objetivo de proporcionar información a los usuarios y permitir transacciones como inserción, eliminación y actualización de datos." (Benítez & Arias, 2017)

El diccionario de la Real Academia Española define una base de datos como el "conjunto de datos organizados de tal modo que permita obtener con rapidez diversos tipos de información." (Real Academia Española, 2001)

1.3.1. Sistemas Gestores de Bases de Datos (SGBD)

Debido a que la información debe ser administrada, mantenida y procesada, muchas veces es necesario el uso de un sistema de gestión bases de datos (SGBD), un conjunto de programas informáticos que permiten gestionar una base de datos. Estos programas especializados no solo permiten manipular la información almacenada, sino que también sirven como una interfaz estándar para los usuarios y otras aplicaciones que la utilizan.

Como ejemplos de los principales sistemas de gestión de bases de datos se encuentran Oracle, SQL Server, Postgres, MySQL, entre otros. (Benítez & Arias, 2017)

1.3.2. Base de Datos Relacional

Uno de los modelos de base de datos más utilizados en la actualidad es el modelo relacional. Propuesto en 1970 por Edgar Frank Codd en IBM, este modelo se basa en teoría de conjuntos y lógica de predicados. En las bases de datos relacionales se utilizan 3 elementos para la representación de sus datos (Nevado Cabello, s.f.):

- Tabla o relación: es la estructura básica de este modelo. Todos los datos de la base de datos se representan en forma de una tabla, conformada por filas y columnas.
- Columnas o atributos: almacenan información sobre una propiedad determinada de la tabla.
- Filas o tuplas: almacenan los valores que toma cada atributo para cada instancia de la relación.

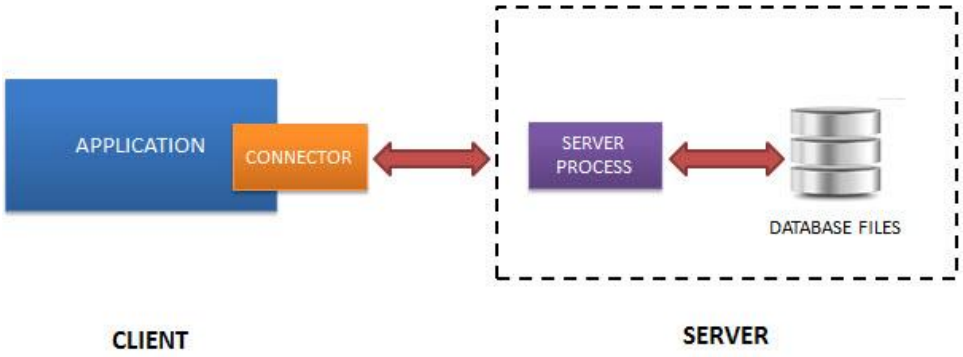
1.3.3. SQLite

"Un SGDB de código abierto cuya principal diferencia es que en lugar de ser un SGDB estándar, está contenido en una biblioteca escrita en lenguaje C." (Benítez & Arias, 2017) SQLite fue diseñado y publicado en el año 2000 por D. Richard Hipp, utilizando como base GDBM, la biblioteca de gestión de bases de datos GNU.

SQLite presenta una gran cantidad de características que pueden ser grandes ventajas, dependiendo de su uso. Entre las principales se encuentra las siguientes:

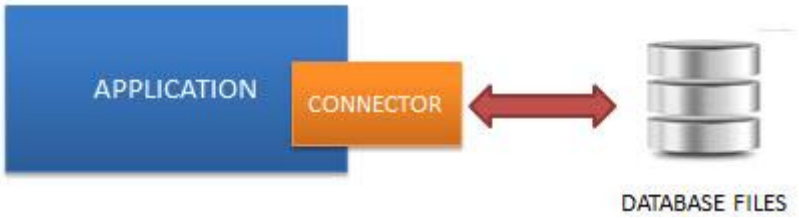
- Al ser una biblioteca y no un programa, SQLite no requiere instalación, si no que basta con integrarlo de la misma manera que se lo haría con una biblioteca normal.
- Es multiplataforma.
- Su tamaño es relativamente pequeño (entre 250kB y 400kB, dependiendo las características y configuraciones utilizadas).
- Sus transacciones cumplen las características ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad), permitiendo la concurrencia desde múltiples procesos o hilos de ejecución.
- Se encuentra embebido en el mismo programa, por lo que no utiliza un modelo de cliente-servidor; no requiere un proceso de servidor aparte.

Ilustración 1.3 – Arquitectura cliente/servidor convencional



SQLite Tutorial. (s.f.). *What is SQLite.* Recuperado de http://www.sqlitetutorial.net/what-is-sqlite/

Ilustración 1.4 – Arquitectura sin servidor de SQLite



SQLite Tutorial. (s.f.). *What is SQLite.* Recuperado de http://www.sqlitetutorial.net/what-is-sqlite/

1.4. Lenguajes de Programación

Un lenguaje de programación es aquel que permite dar instrucciones o acciones consecutivas a un ordenador, posibilitando la creación de procesos, algoritmos y programas que pueden ser ejecutados por máquinas. Al igual que cualquier lenguaje, su estructura se basa en símbolos y reglas sintácticas y semánticas.

El diccionario de la Real Academia Española define un lenguaje de programación como el "conjunto de signos y reglas que permite la comunicación con un ordenador." (Real Academia Española, 2001)

Se dice que un lenguaje es de más bajo nivel cuando se asemeja al lenguaje de máquina, es decir, el binario. Debido a que este lenguaje está compuesto por cadenas de 0s y 1s, no es comprensible para los seres humanos, lo cual llevó a la necesidad de crear lenguajes

intermedios de mayor nivel; es decir, lenguajes que se asemejen más a nuestro lenguaje natural o humano. El ensamblador fue el primer lenguaje de programación utilizado. Su nivel es inmediatamente superior al de lenguaje de máquina, lo que lo hace similar a éste, pero posee unas pequeñas modificaciones mnemotécnicas que facilitan a los expertos su comprensión y uso. (CMM, 2017)

1.4.1. Java

"Lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible." (Abellán Martínez, 2015)

Java presenta las siguientes ventajas:

- Es un lenguaje orientado a objetos.
- Es multiplataforma
- Fue diseñado para ser fácil de aprender y mantener una sintaxis simplifica.
- Es seguro, proveyendo características que permiten mantener los programas libres de virus.
- El código se ejecuta mediante una máquina virtual, permitiendo la independencia de la arquitectura de la máquina que lo compila o ejecuta.
- Es robusto, proporcionando numerosas comprobaciones en compilación y en tiempo de ejecución.
- Soporta *multithreading*, o múltiples hilos de ejecución.
- Es un lenguaje interpretado.
- Es de alto rendimiento.
- Está diseñado para entornos distribuidos.
- El lenguaje y su sistema de ejecución en tiempo real son dinámicos, permitiendo su adaptación a diversos ambientes.

El proyecto del lenguaje Java inició en 1991 por James Gosling, de la compañía Sun Microsystems, como una herramienta de programación. En 1995 fue publicado como un componente de la plataforma de Java. 11 años después, Sun Microsystems lanzó la versión 6 de Java como libre y de código abierto, bajo la Licencia Publica General de GNU (GNU GPL). (Tutorials Point, s.f.)

A finales del año 2009, Oracle Corporation adquirió Sun Microsystems, incluyendo Java, por un montante estimado de 5.600 millones de dólares, que alcanzaba los 7.400 millones de dólares al sumarle la deuda de la empresa adquirida. (El País, 2009)

Dicha adquisición generó muchas dudas, preocupaciones y desacuerdos, tanto entre la comunidad como entre los empleados de Sun Microsystems, ya que esta empresa era uno de los mayores competidores de Oracle y se temía la posibilidad de que se comercialice Java. En el transcurso del siguiente año, varios ingenieros y personal renombrado del equipo renunciaron debido a la nueva adquisición, entre ellos James Gosling (el creador de Java), Tim Bray (creador de XML), Kohsuke Kawaguchi (líder de desarrollo de la herramienta Hudson), Bran Cantrill (co-creador del framework DTrace), y Jonathan Schwartz (director ejecutivo de Sun Microsystems). (De Seta, 2010)

Desde sus inicios, Java se volvió uno de los lenguajes de programación más populares, y actualmente se mantiene como el lenguaje más utilizado.

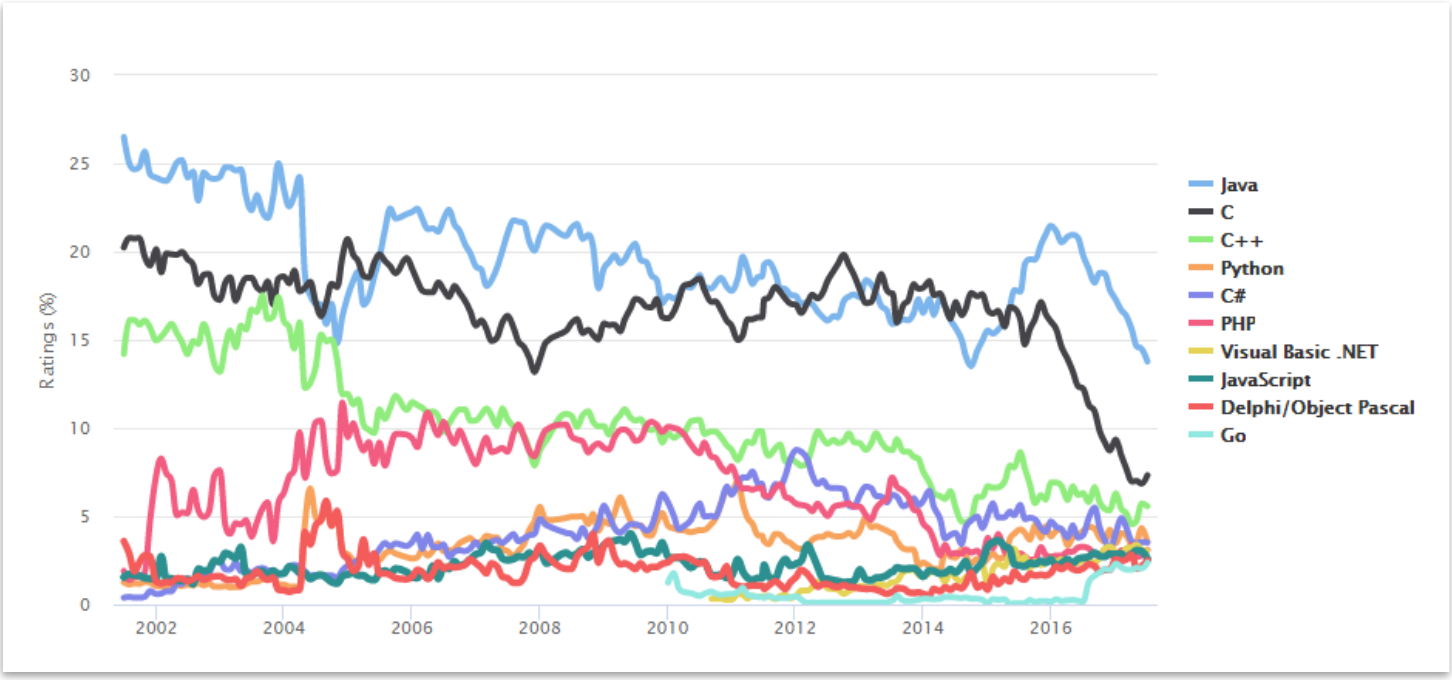
Ilustración 1.5 – Índice de popularidad de lenguajes de programación para julio 2017
Resultados en base a búsquedas de tutoriales de los lenguajes de programación

Worldwide, Jul 2017 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Java	22.6 %	-1.1 %
2		Python	16.4 %	+4.0 %
3		PHP	9.1 %	-1.0 %
4		C#	8.2 %	-0.6 %
5		Javascript	8.0 %	+0.6 %
6		C++	6.6 %	-0.3 %
7		C	6.5 %	-0.3 %
8	↑	R	3.7 %	+0.5 %
9	↓	Objective-C	3.6 %	-1.1 %
10		Swift	2.8 %	-0.3 %

PopularitY of Programming Language. (2017, julio). Recuperado en julio 17, 2017, de http://pypl.github.io/PYPL.html

Ilustración 1.6 – Índice TIOBE de popularidad de lenguajes de programación para julio 2017
Resultados en base al número de ingenieros calificados en el mundo, cursos, proveedores y búsquedas en los principales motores de búsqueda



TIOBE. (2017, julio). Recuperado en julio 17, 2017, de https://www.tiobe.com/tiobe-index/

1.4.2. C#

C# (pronunciado en inglés, C Sharp), es un lenguaje de programación simple, moderno, de propósito general y orientado a objetos. Fue desarrollado por Anders Heljsberg y su equipo de Microsoft como parte de la plataforma .NET, en el año 1999. Al igual que Java, su sintaxis fue construida basándose en C/C++.

"C# es un lenguaje de programación que se ha diseñado para compilar diversas aplicaciones que se ejecutan en .NET Framework. C# es simple, eficaz, con seguridad de tipos y orientado a objetos. Las numerosas innovaciones de C# permiten desarrollar aplicaciones rápidamente y mantener la expresividad y elegancia de los lenguajes de estilo de C." (Microsoft, 2017)

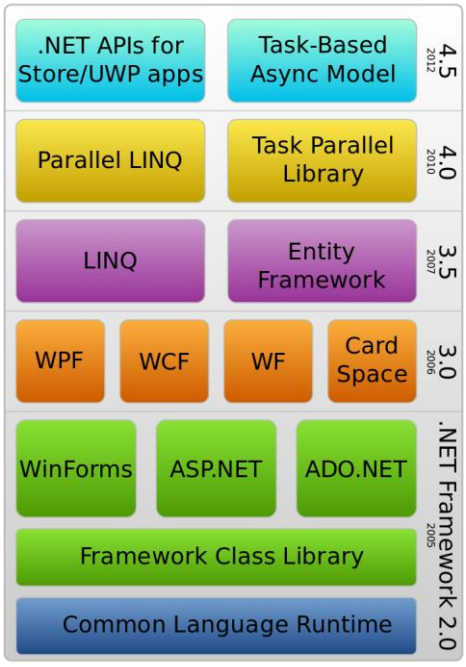
1.5. .NET Framework

.NET es un framework desarrollado por Microsoft, que busca ofrecer una manera rápida y a la vez económica de desarrollar aplicaciones, sin dejar de lado la seguridad y la robustez. Microsoft asegura que su framework provee un modelo de programación comprensivo para desarrollar todo tipo de aplicaciones para Windows, sea móvil, web o de escritorio. (Microsoft, s.f.)

Entre las principales herramientas que proporciona el framework .NET se encuentran:

- Framework Class Library (FCL): una biblioteca que provee clases de interfaces de usuario, conectividad a base de datos, desarrollo de aplicaciones web, entre otras. Forma la base en la que las aplicaciones, los controles y los componentes serán construidos en .NET.
- Common Language Runtime (CLR): una máquina virtual que provee servicios, entre sus principales servicios de seguridad, administración de memoria, manejo de excepciones, administración de hilos y recolección de basura.

Ilustración 1.7 – Componentes de .NET Framework



Soumyasch. (2007, octubre). Recuperado en julio 19, 2017, de https://en.wikipedia.org/wiki/.NET_Framework

1.5.1. ADO.NET

ADO.NET Entity Framework es un conjunto de componentes que permite el acceso a los datos y servicios de datos de una base de datos, relacional o no relacional, permitiendo "crear aplicaciones de acceso a datos programando con un modelo de aplicaciones conceptuales en lugar de programar directamente con un esquema de almacenamiento relacional. El objetivo es reducir la cantidad de código y el mantenimiento necesarios para las aplicaciones orientadas a datos." (Microsoft, s.f.)

Este conjunto de componentes permite el desarrollo de aplicaciones que sean libres de dependencias de codificación rígida de un motor de datos o de un esquema de almacenamiento, donde la capa de acceso a los datos se encuentra separada de la capa de su manipulación. ADO.NET permite el uso de LINQ, de manera que se proporcione una validación en el momento de compilación para las consultas en un modelo conceptual.

1.5.2. LINQ

Language Integrated Query (traducido como Consulta Integrada del Lenguaje) es un componente de la plataforma .NET, que permite agregar a los lenguajes de .NET, de manera nativa, capacidades de consultas a datos de diversas fuentes y formatos. Utilizando un conjunto de reglas de traducción, el compilador traduce un conjunto definido de nombres de métodos, expresiones lambda y tipos anónimos en expresiones de consulta normales. Este componente fue lanzado en noviembre del 2007, como parte principal de .NET Framework 3.5.

1.5.3. Entity Framework

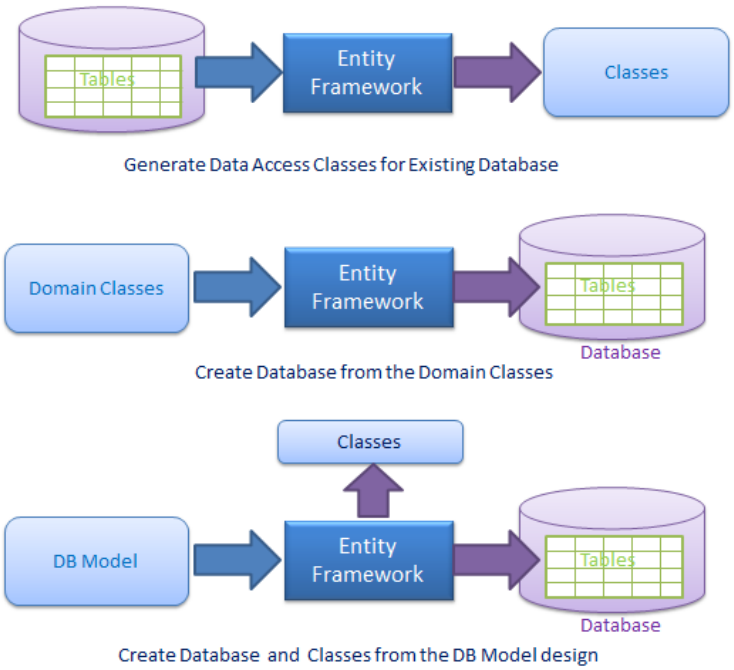
Entity Framework es un conjunto de tecnologías desarrollado por Microsoft para la plataforma .NET, específicamente para ADO.NET, para el mapeo objeto-relacional. Este framework busca eliminar la mayoría de las codificaciones que necesitan realizarse para acceder a la data.

Entity Framework resulta especialmente útil cuando:

- Se tiene creada la base de datos, y se quiere crear las clases de dominio a partir de esta.
- Se tienen creadas las clases de dominio, y se quiere crear la base de datos a partir de estas.

- Se tiene diseñado el esquema de la base de datos, y se quiere crear la base de datos y las clases de dominio a partir de este.

Ilustración 1.8 – Escenarios de uso de .NET Framework



Entity Framework Tutorial. (s.f.). Recuperado de http://www.entityframeworktutorial.net/what-is-entityframework.aspx

1.6. IDE

Un entorno de desarrollo integrado, o IDE (por sus siglas en inglés, Integrated Development Environment), "es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI)." (García, 2013) Los IDEs también suelen incluir otras herramientas que facilitan a los desarrolladores la escritura, diseño, modificación, despliegue y depuración de programas, mejorando su productividad e incluso la velocidad de aprendizaje de un nuevo lenguaje de programación o biblioteca.

1.6.1. Android Studio

Android Studio es el IDE oficial para la plataforma Android, desarrollado por Google y basado en el entorno de desarrollo de JetBrains, IntelliJ IDEA. Fue liberado en etapa beta en la versión 0.8, en junio del 2014, reemplazando a Eclipse como el IDE oficial para el desarrollo de aplicaciones Android. En diciembre del mismo año, se lanzó la versión 1.0, la primera versión estable. Android Studio 2.3.3 es la última versión estable al momento, liberada en junio del 2017, y está disponible para Windows (32bits y 64 bits), macOS y GNU/Linux.

Entre las principales características que se encuentran en la última versión de Android Studio, están (Desde Linux, 2016):

- Renderizado en tiempo real.
- Soporte para construcción basado en Gradle.
- Constructor de interfaz gráfica que permite a los usuarios arrastrar y soltar los componentes de la interfaz de usuario.
- Herramientas "Lint", para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones, entre otros.
- Asistentes para la creación de diseños y otros componentes basados en plantillas.
- Soporte para el desarrollo de aplicaciones para Android Wear, Android TV, Android Auto y Android Glass.
- Administrador de dispositivos virtuales de Android, permitiendo probar y ejecutar las aplicaciones en diferentes modelos y versiones de Android.

Android Studio también se encarga de proveer el kit de desarrollo de software, Android SDK, el cual se encargará del correcto funcionamiento de las aplicaciones en dispositivos Android. El lenguaje de programación que Android Studio utiliza para desarrollar es Java, aunque es posible agregar código C y C++ a la aplicación mediante la instalación de un complemento, disponible para las versiones 2.2 y posteriores. (Android Studio, s.f.)

"Android Studio hace la vida mucho más fácil en comparación con software no especializado, pero todavía tiene un poco más de camino por recorrer antes de que pueda pretender ser una experiencia completamente intuitiva y fluida." (Mullis, 2017)

1.6.2. Visual Studio

Visual Studio es el IDE de Microsoft para el diseño, desarrollo, depuración e implementación de programas, sitios web, aplicaciones web, servicios web y aplicaciones móviles. Su editor de texto y depurador soportan casi cualquier lenguaje de programación, teniendo incorporados lenguajes tales como C, C++, VB.NET, C#, F#, TypeScript, XML, HTML/XHTML, JavaScript y CSS, y permitiendo la instalación de servicios para trabajar con otros lenguajes. También cuenta con herramientas y componentes enfocados en una variedad de aspectos, como el trabajo en equipo, el manejo de ciclos de vida de software, el control de versiones, entre otros.

La primera versión, Visual Studio 97, fue liberada en febrero de 1997, en un intento de Microsoft de usar un mismo entorno de desarrollo para múltiples lenguajes. Visual Studio 2017, la versión 15.0, es la última versión estable, liberada el 7 de marzo del 2017.

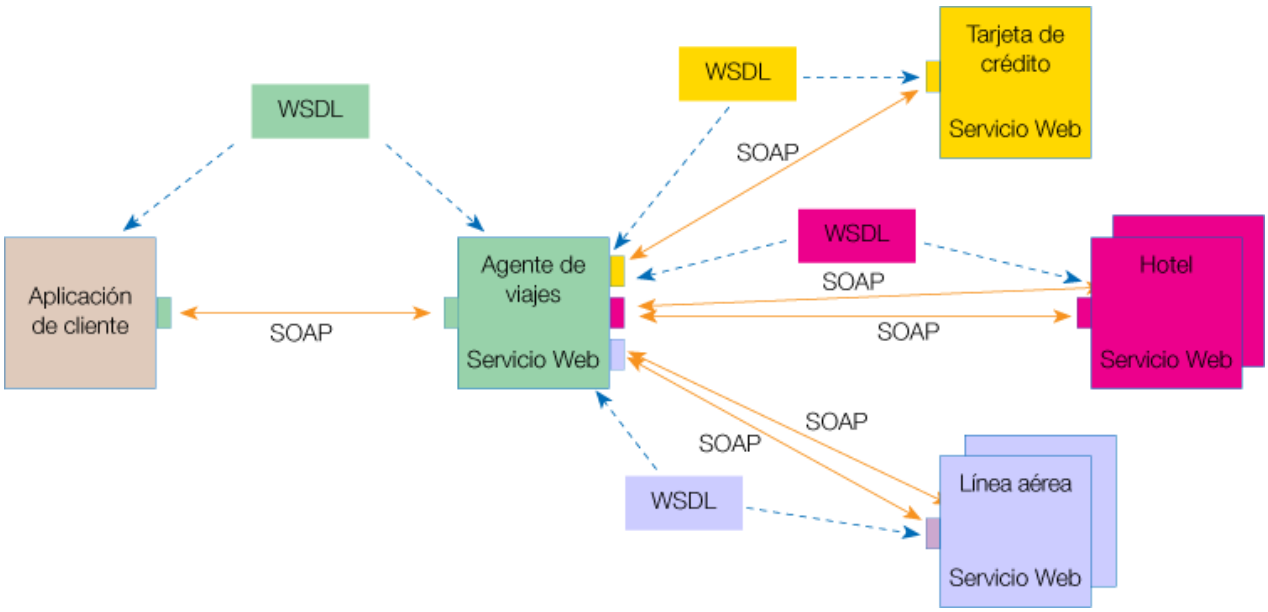
Microsoft maneja las siguientes 4 ediciones para su IDE (Microsoft, s.f.):

- Express edition: lanzada por primera vez con Visual Studio 2005, provee herramientas y complementos limitados y enfocados hacia un tipo específico de desarrollo (por ejemplo, aplicaciones Web, o de escritorio). Esta edición gratuita fue pensada para estudiantes y desarrolladores informales. La última versión de Visual Studio dejó de ofrecer esta edición, a cambio de la Community edition.
- Community edition: apareció en noviembre del 2014 como la primera edición gratuita no limitada de Visual Studio (excluyendo versiones de prueba). A partir de entonces se ha mantenido esta edición, que incluye la mayor parte de características de Visual Studio, pero es orientada a grupos de desarrollo de una o pocas personas.
- Professional edition: versión de nivel comercial de Visual Studio, que provee herramientas de trabajo en equipo, como Team Foundation Server, y soporte para Microsoft Developer Network (MSDN).
- Enterprise edition: la versión más completa de Visual Studio, con soporte completo para todas las herramientas disponibles para desarrollo, pruebas y depuración de software, manejo y validación de arquitecturas, diagnósticos avanzados, desarrollo con base de datos, desarrollo multiplataforma, entre otros.

1.7. Servicios Web

Un servicio web (o web service, en inglés) puede definirse como un conjunto de tecnologías que sirven para el intercambio de datos a través de la red, mediante el uso de un conjunto de protocolos y estándares. Proporcionan interoperabilidad, permitiendo que diferentes aplicaciones, que pueden estar escritas en diferentes lenguajes de programación y operar en diferentes dispositivos, puedan interactuar entre sí. De esta manera, servicios y aplicaciones de diversas empresas pueden ser fácilmente integrados.

Ilustración 1.9 – Ejemplo de servicios web en funcionamiento



Entity Framework Tutorial. (s.f.). Recuperado de http://www.entityframeworktutorial.net/what-is-entityframework.aspx

"Los servicios web son aplicaciones modulares autodescriptivas que se pueden publicar, ubicar e invocar desde cualquier punto de la red o desde el interior de una red local, basados en estándares abiertos de Internet. Ya no es necesario que el proveedor y el usuario de un servicio web tengan el mismo sistema operativo y utilicen el mismo lenguaje de programación, dado que se basan en estándares aceptados plenamente por la industria, como XML, HTTP y SMTP." (García Valcárcel & Munilla Calvo, 2003)

Los servicios web se componen principalmente de 4 capas (Lamarca Puente, 2013):

- Capa de transporte, donde se encuentran los servicios de transporte que se encargan de establecer la conexión. Los ejemplos más comunes son HTTP, SMTP y FTP.
- Capa de mensajería, donde se encuentran los servicios de mensajería que especifican cómo se codificarán los mensajes que se intercambiarán entre el cliente y el servidor. Los ejemplos más comunes son los protocolos SOAP y REST.
- Capa de descripción, donde se ofrece un formato básico de la descripción de las peticiones del servicio web. Se especifica el formato que se debe usar para comunicarse con el servicio web y otra información, como la dirección. El lenguaje WSDL permite realizar dichas descripciones.
- Capa de descubrimiento, donde se encuentran los servicios que permiten a los clientes encontrar de forma dinámica los servicios web. El estándar UDDI es un ejemplo utilizado en esta capa.

La información se transfiere mediante formatos que, si bien suelen ser basados en texto y son entendibles por humanos, también son entendibles entre máquinas. Los principales ejemplos de estos son (R., 2014):

- XML (eXtensible Markup Language): lenguaje de marcado desarrollado por la W3C, que permite representar cualquier estructura de datos. Sus mayores fortalezas son el amplio soporte con el que cuenta en la actualidad y su manejo de caracteres Unicode.
- JSON (JavaScript Object Notation): formato ligero que representa la información como una colección de pares de atributo/valor, a manera de texto plano, permitiendo codificar con UTF-8, UTF-16 o UTF-32. Su simplicidad y facilidad de implementación le otorgan un gran desempeño. Actualmente es utilizado como una de las principales alternativas de XML.

1.7.1. SOAP

El protocolo de acceso a objetos simples, o SOAP (por sus siglas en inglés, Simple Object Access Protocol), es un protocolo simple que define una organización para el intercambio de información estructurada. Utiliza tecnologías XML para definir un marco extensible de mensajería, proveyendo una construcción del mensaje que se puede intercambiar sobre una variedad de protocolos subyacentes. Este marco está diseñado para ser independiente

de cualquier modelo de programación y de cualquier otra semántica específica de la implementación. (W3C, 2007)

1.7.2. REST

La transferencia de estado representacional, o REST (por sus siglas en inglés, REpresentational State Transfer), es un estilo de arquitectura que hace uso del protocolo HTTP. Proporciona una interfaz, o API, que permite utilizar cada uno de sus métodos (siendo GET, POST, PUT y DELETE los más comunes) para poder realizar diferentes operaciones entre la aplicación que ofrece el servicio web y el cliente.

"Si bien el término REST se refería originalmente a un conjunto de principios de arquitectura, en la actualidad se usa en el sentido más amplio para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc.) sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como por ejemplo SOAP." (Sánchez Tapia, 2013)

REST define que la comunicación entre el cliente y el servidor es sin estado; el contexto del cliente no se almacena en el servidor, por lo que cada mensaje contiene toda la información necesaria para comprender la petición. Asimismo, REST se basa en recursos que pueden ser accedidos a través de una URI.

1.7.3. SOAP vs REST

Tanto SOAP como REST son protocolos de comunicación de servicios web escalables. Si bien ambos pueden llegar a lograr el mismo resultado en la mayoría de los casos, existen algunas diferencias que los vuelven más aptos, dependiendo de la situación. (Stringfellow, 2017)

El protocolo SOAP solo permite el uso de XML para la transmisión de datos, mientras que REST ofrece una mayor flexibilidad, permitiendo alternativas como JSON, que pueden llegar a trabajar de manera más rápida, sencilla y consumiendo menos ancho de banda. En el caso de una página web funcional, integrar servicios REST resulta menos complejo, evitando tener que reestructurar la infraestructura de la página web. Los servicios web con REST están enfocados en la exposición de información, mientras que los servicios con SOAP se enfocan en exponer operaciones.

Por otro lado, existen situaciones donde el uso de SOAP resultará más provechoso. SOAP provee más seguridades para la privacidad y la integridad de la información, resultando más apto para servicios web que requieren una seguridad robusta. SOAP también provee una lógica de reintento integrada cuando la comunicación falla, algo que debe implementarse manualmente al utilizar servicios REST. Otra cualidad de SOAP que puede resultar en una ventaja es el uso de un set de reglas estandarizados, que permite que un proveedor y un cliente puedan comunicarse sin la necesidad de establecer reglas para el contenido y el contexto de los mensajes.

1.8. Metodologías de Desarrollo

Una metodología de desarrollo de software se refiere al marco de trabajo que se usa para estructurar, planificar, y controlar el proceso de desarrollo de un sistema de información. A lo largo de los años, una gran variedad de metodologías de desarrollo han aparecido y evolucionado, cada una con sus fortalezas y debilidades. Considerando varios aspectos técnicos, organizacionales, del proyecto y del equipo de desarrollo, algunas metodologías se adaptarán mejor a distintos proyectos, mientras que otras podrán no adaptarse del todo. (CMS, 2005)

1.8.1. Extreme Programming

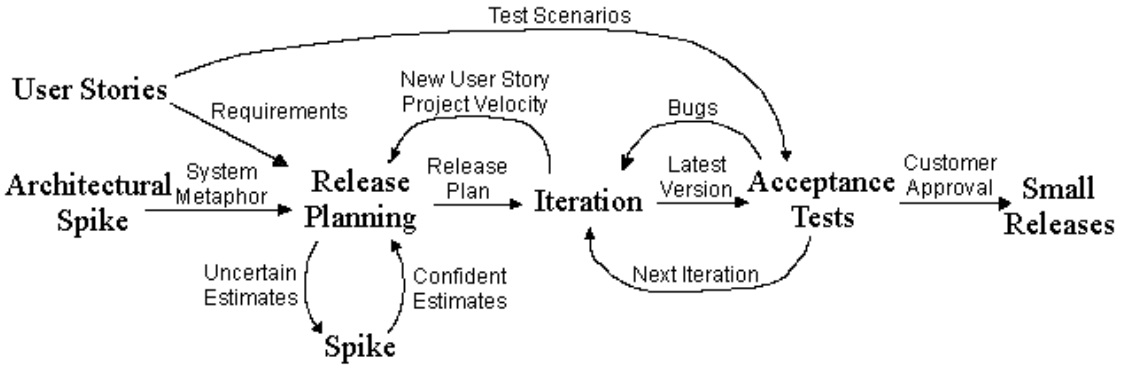
La programación extrema, o XP (por sus siglas en inglés, eXtreme Programming), es una metodología de desarrollo ágil definida por Kent Beck, en 1999. XP se enfoca en la satisfacción del cliente mediante el desarrollo frecuente y rápido de entregables de software, a medida que se vayan necesitando. Esta metodología crea un ambiente simple, pero efectivo, que busca incrementar la productividad y la autoorganización del equipo. De igual manera, permite a los desarrolladores responder con confianza a los cambios que pueden presentarse en los requerimientos, incluso cuando el proyecto se encuentra en una etapa avanzada. XP promueve la práctica de la programación en parejas, la cual permite a los programadores revisar y discutir el código a medida que se escribe.

XP busca la mejora del proyecto en 5 aspectos principales (Wells, 2013):

- La comunicación con el cliente, de manera que se puedan implementar los cambios sugeridos, y entre los desarrolladores.
- La simplicidad en el diseño.
- La retroalimentación, mediante pruebas constantes que se mantienen desde el inicio.
- El respeto por cada miembro del equipo, el cual se profundiza con cada pequeño éxito.
- El coraje con el cual se debe afrontar los cambios de requerimientos y de tecnología.

El desarrollo mediante esta metodología comienza con el levantamiento de requerimientos a manera de historias de usuario, es decir, descripciones de las funciones que el usuario necesita en el sistema, sin limitarse necesariamente a la interfaz de usuario. Estas deben ser escritas por el cliente, manteniendo su perspectiva y un lenguaje sencillo y libre de detalles técnicos. Las historias de usuario consisten en escenarios descritos en aproximadamente 3 oraciones, donde se incluye el rol del usuario, el objetivo de la funcionalidad, y su finalidad o beneficio. Su nivel de detalle debe ser solamente el suficiente como para permitir realizar un cálculo de bajo riesgo sobre el tiempo que tardará su implementación. La descripción detallada se obtendrá del cliente una vez que la historia de usuario vaya a ser implementada.

Ilustración 1.10 – Desarrollo de un proyecto con Extreme Programming

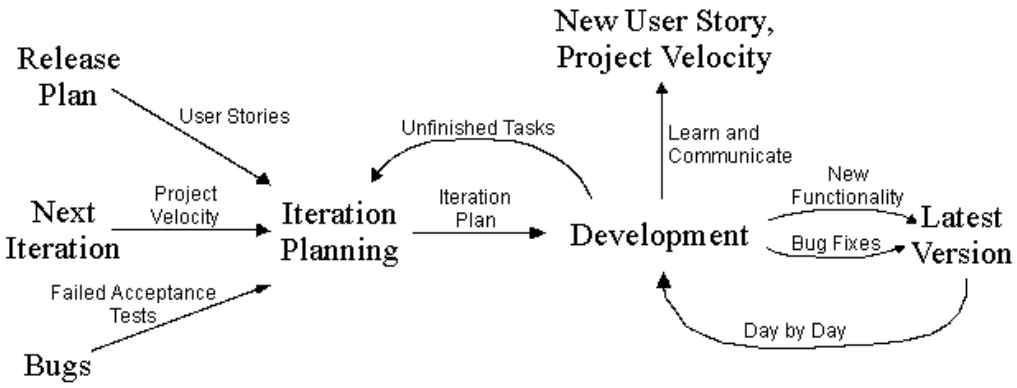


Wells, Don. (2000). Recuperado de http://www.extremeprogramming.org/map/project.html

Una vez que se tienen las historias de usuario, se procede a realizar un release planning (planificación de publicaciones), donde tanto desarrolladores como clientes tomarán decisiones en la planificación del proyecto, tomando en cuenta el alcance, los recursos, el tiempo y la calidad del proyecto. Se establecen los tiempos óptimos de desarrollo de las historias de usuario, es decir, el tiempo que tomará su desarrollo y pruebas correspondientes, considerando que no existirán impedimentos o retrasos. Las historias de usuarios son ordenadas por prioridad, y agrupadas en versiones del programa, que serán desarrolladas de manera iterativa.

A continuación, se procede a realizar las iteraciones, las cuales deben durar entre 1 y 3 semanas. Primero, se mantienen una planificación de la iteración, donde se seleccionarán las historias de usuario a implementarse del plan de liberación, seleccionando también, en el caso de haber, aquellas que hayan fallado las pruebas de aceptación en la anterior iteración. Las pruebas de aceptación de cada historia de usuario deben estar definidas al inicio de esta fase. Para su implementación, las historias de usuario se dividen en tareas detalladas, con un tiempo óptimo de desarrollo de 1, 2 o 3 días. Posteriormente, los desarrolladores seleccionan las tareas, responsabilizándose de su implementación y pruebas correspondientes. Es importante tomar en cuenta la velocidad de avance del proyecto de las iteraciones anteriores, con el fin de no excederla.

Ilustración 1.11 – Iteración en un proyecto con Extreme Programming



Wells, Don. (2000). Recuperado de
http://www.extremeprogramming.org/map/iteration.html

Durante el desarrollo del programa, la metodología XP fomenta la programación en parejas, pues asegura que esta práctica incrementa la calidad del software, manteniendo la misma funcionalidad y el mismo tiempo de desarrollo que si los programadores trabajarían de manera independiente. La refactorización al utilizar código antiguo debe ser algo que sea realice frecuentemente, con el fin de pulir y optimizar el diseño del programa, y mejorar su mantenibilidad. Es necesario mantener reuniones diarias con el equipo de desarrollo, las cuales deben ser rápidas y fluidas, sin extenderse más de 15 minutos. En ellas, cada desarrollador debe comunicar sobre lo que realizó el día anterior, lo que realizará ese día, y los problemas o inconvenientes que tenga y puedan causar demoras.

Finalmente, antes de liberar una versión, las historias de usuarios deben pasar sus pruebas de aceptación. Como su nombre lo indica, estas son pruebas de caja negra, que sirven como criterio de aceptación, asegurando que la implementación ha sido correcta. Cada historia de usuario puede tener una o varias pruebas de aceptación, de manera que se asegure la correcta verificación de su funcionalidad. Las pruebas de aceptación se realizan y verifican con el cliente al final de la iteración; sin embargo, deben seguirse verificando en las siguientes iteraciones, para verificar que las versiones anteriores sigan funcionando. Es por esto que las pruebas de aceptación deben ser automatizadas.

2. Requerimientos

2.1. Diagnóstico de la Situación Actual

El Museo QCAZ de la Pontificia Universidad Católica del Ecuador maneja los datos de las colecciones biológicas. Con el transcurso de los años, el Museo QCAZ ha recopilado miles de registros de información acerca de la rama de herpetología, que encierra a las especies de anfibios y reptiles del Ecuador, lo cual ha requerido la inversión de una gran cantidad de recursos. El incremento de la información ha permitido la creación de una página web que trabaja en conjunto con la base de datos, la cual es utilizada por investigadores no solo de Ecuador y Latinoamérica, sino incluso de países como Estados Unidos y España.

Actualmente, se está trabajando en un proyecto que consiste en la creación de una nueva plataforma tecnológica para administrar y difundir la información con herramientas de última tecnología, con lo que se busca solucionar diversos problemas que se han presentado previamente y mejorar sus funcionalidades.

Junto con la creación de esta nueva plataforma, también se desea permitir el almacenamiento y acceso, a través de un aplicativo móvil con conexión en línea, a los cuadernos de campo, una herramienta usada por investigadores de diversas áreas, entre ellas la biología, que consiste en un cuaderno o bloc de notas, donde se escriben o dibujan apuntes y observaciones cuando se realizan trabajos en el campo. "Tras servir para la elaboración de investigaciones, los cuadernos de campo se transforman en documentos, generalmente presentes en los archivos personales de los académicos y, dependiendo del valor histórico de sus obras, pueden pasar a integrar archivos públicos". (Roa & Vargas, 2009) Dichos cuadernos de campo son una herramienta fundamental para las investigaciones que se llevan a cabo en el museo QCAZ, pues brindan información que posteriormente darán lugar a observaciones que pueden sustentar o invalidar argumentaciones propuestas.

Este aplicativo móvil busca ayudar a los estudiantes de la Pontificia Universidad Católica del Ecuador, y a otros investigadores involucrados, a tomar apuntes en sus salidas de campo, mejorando la protección de los datos obtenidos y la automatización de su almacenamiento. También permite al investigador aprovechar diversas herramientas y sensores de su dispositivo móvil para obtener información útil de su entorno, como por ejemplo la ubicación, la temperatura y la humedad.

2.2. Levantamiento de Requerimientos

La metodología XP requiere que el levantamiento de requerimientos se lo realice en forma de historias de usuario, pues estas son más que simples documentos de requerimientos, que luego servirán para realizar pruebas automatizadas que verifiquen el correcto funcionamiento del sistema.

Para levantar los requerimientos del proyecto, se mantuvo una reunión con el Msc. Damián Nicolalde, director de Bioinformática de la facultad de Biología de la PUCE, y en base a esta reunión se levantaron las siguientes historias de usuario:

	Enunciado de la Historia		
Identificador de la Historia	Rol	Característica / Funcionalidad	Razón / Resultado
US-1	como un usuario registrado	necesito poder validar mis credenciales	con la finalidad de poder ingresar al sistema

Tabla 2.1 – Historia de usuario US1

	Enunciado de la Historia		
Identificador de la Historia	Rol	Característica / Funcionalidad	Razón / Resultado
US-2	como un usuario registrado	necesito poder poder ingresar la información de un cuaderno de campo	para poder registrarlo

Tabla 2.2 – Historia de usuario US2

	Enunciado de la Historia		
Identificador de la Historia	Rol	Característica / Funcionalidad	Razón / Resultado
US-3	como un usuario registrado	necesito poder obtener la ubicación, la temperatura y la humedad	para poder registrarlo en el cuaderno de campo en caso de ser necesario

Tabla 2.3– Historia de usuario US3

	Enunciado de la Historia		
Identificador de la Historia	Rol	Característica / Funcionalidad	Razón / Resultado
US-4	como un usuario registrado	necesito poder ver los cuadernos de campo registrados	para poder revisar su información

Tabla 2.4 – Historia de usuario US4

	Enunciado de la Historia		
Identificador de la Historia	Rol	Característica / Funcionalidad	Razón / Resultado
US-5	como un usuario registrado	necesito poder ver la información de los cuadernos de campo registrados	para poder revisar y editar la información de los cuadernos de campo, y eliminarlos

Tabla 2.5 – Historia de usuario US5

2.3. Sistema Propuesto

El aplicativo para dispositivos móviles permitirá a los usuarios escribir sus cuadernos de campo. La aplicación aprovechará los sensores y herramientas disponibles en el dispositivo móvil del usuario para obtener la ubicación, la temperatura y la humedad. El aplicativo móvil estará conectado a la base de datos del museo QCAZ de la Pontificia Universidad Católica del Ecuador, permitiendo que los datos ingresados sean transferidos automáticamente y almacenados en dicha base de datos. En caso de que el dispositivo móvil no disponga de conexión a internet, los datos del cuaderno de campo se almacenarán localmente hasta que la conexión esté disponible y pueda sincronizarse la información. El usuario también podrá cargar, modificar y eliminar los cuadernos de campo que haya guardado previamente.

2.3.1. Casos de Uso

2.3.1.1. Caso de Uso General

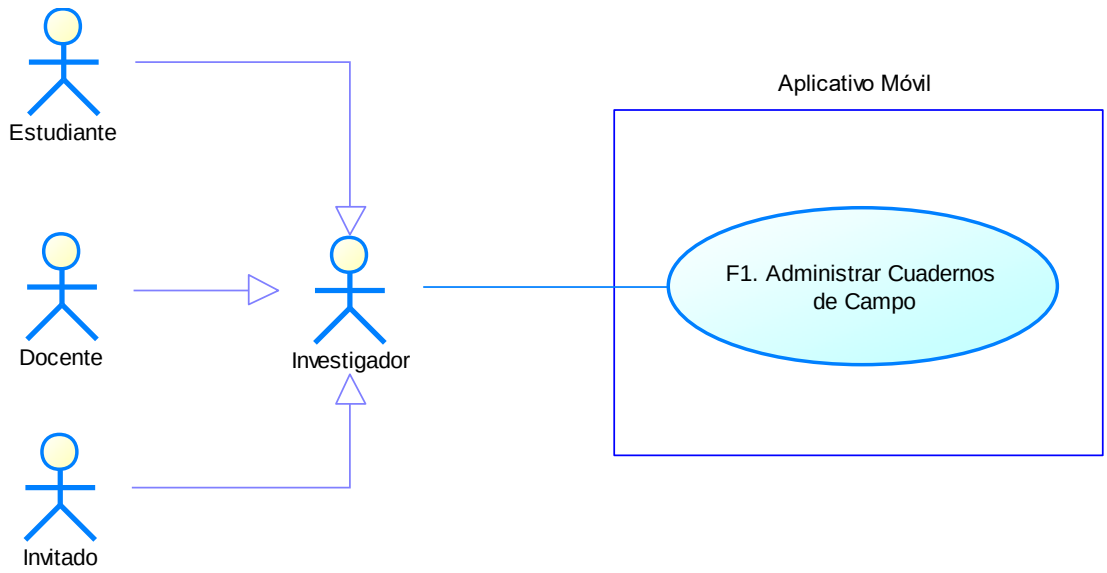


Ilustración 2.1 – Caso de uso general

2.3.1.2. Caso de Uso Específico

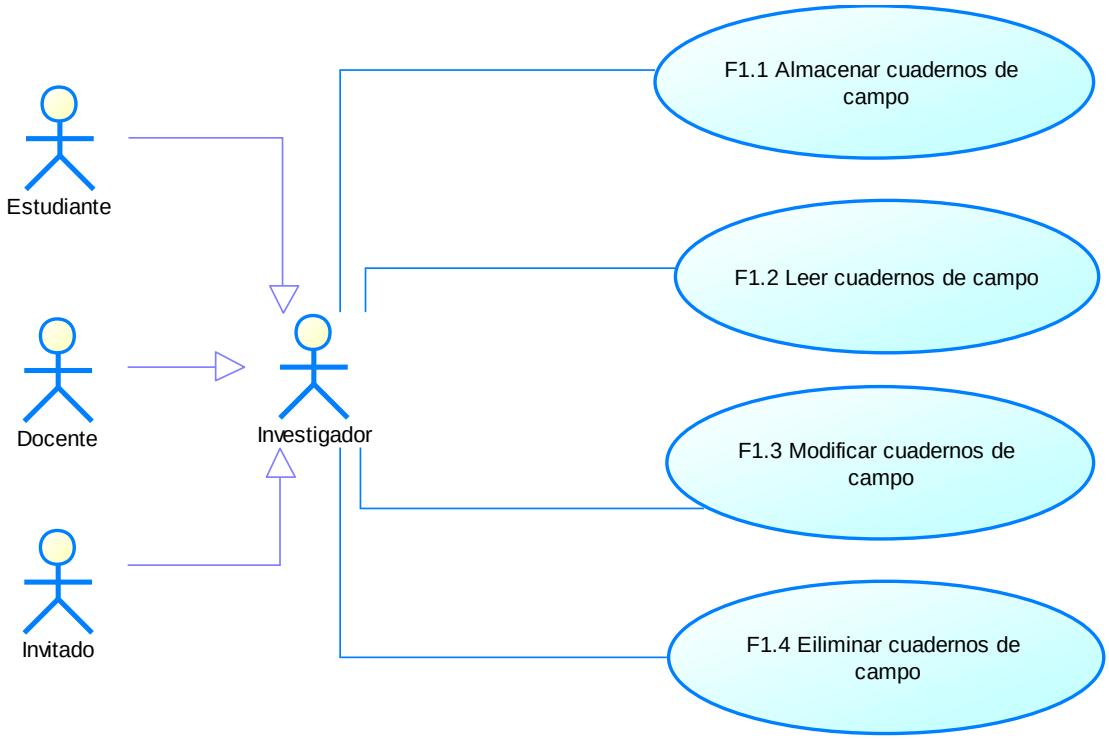
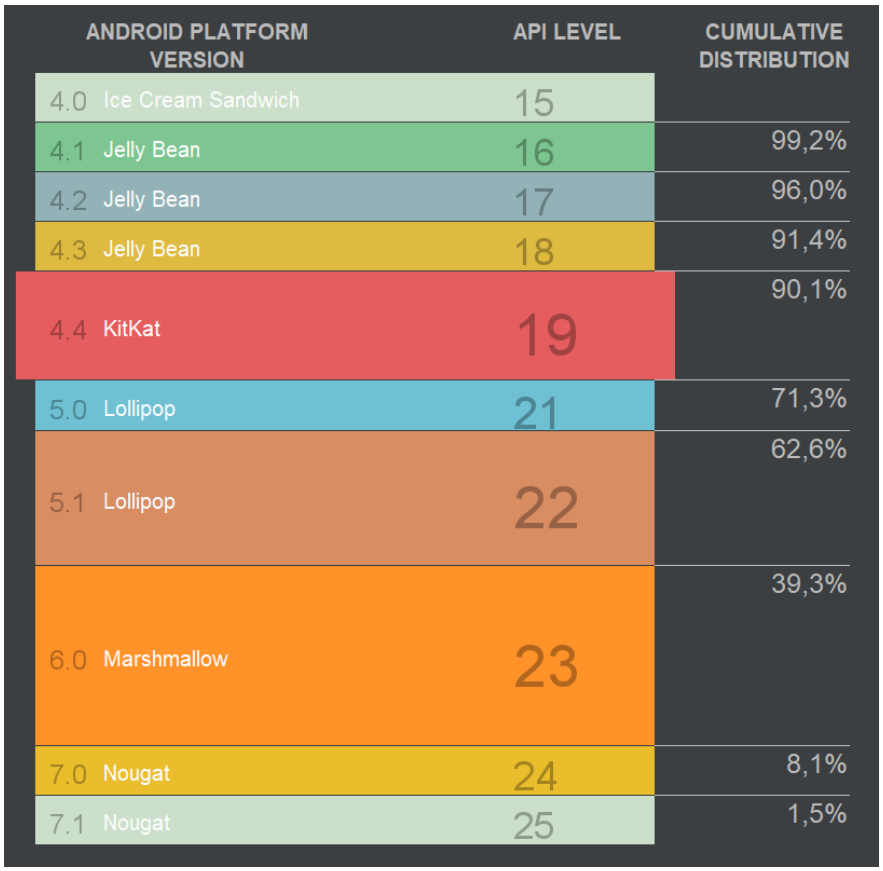


Ilustración 2.2 – Caso de uso específico

2.4. Entorno del Software

- Lenguaje de programación de aplicativo móvil: Java
- Entorno de desarrollo de aplicativo móvil: Android Studio 2.3.3
- SGBD de aplicativo móvil: SQLite, ya que esta pequeña biblioteca cuenta con soporte completo en Android, y se encontrará embebida en el mismo aplicativo.
- Sistema operativo mínimo: Android 4.4 – KitKat (API 19)

Ilustración 2.3 – Distribución de versiones de plataformas/API de Android en el mercado



Recuperado en agosto 31, 2017

- Protocolo de comunicación de servicios web: REST, ya que esta es una manera más rápida, sencilla y que consume menos ancho de banda.
- Lenguaje de programación de servicios web: C#, pues el servidor del museo QCAZ se encuentra levantado en .NET.
- Entorno de desarrollo de servicios web: Visual Studio 2012
- SGBD del museo QCAZ: SQL Server
- Servidor de aplicaciones: Internet Information Services (IIS)

3. Desarrollo

3.1. Primera iteración

3.1.1. Análisis

Historias de usuario: US-1

Duración: 3 semanas

Criterios de aceptación:

Criterios de Aceptación				
Número de Escenario	Criterio de Aceptación	Contexto	Evento	Resultado / Comportamiento Esperado
1	usuario válido con verificación a través de internet	en caso que exista una conexión a internet	cuando el usuario ingrese unas credenciales válidas	permitir al usuario ingresar al sistema
2	usuario no válido con verificación a través de internet	en caso que exista una conexión a internet	cuando el usuario ingrese unas credenciales inválidas	notificar al usuario que las credenciales ingresadas no son válidas
3	usuario válido con verificación local	en caso que no exista una conexión a internet	cuando el usuario ingrese unas credenciales que hayan sido validadas anteriormente	permitir al usuario ingresar al sistema
4	usuario no válido con verificación local	en caso que no exista una conexión a internet	cuando el usuario ingrese unas credenciales que no hayan sido validadas anteriormente	notificar al usuario que las credenciales ingresadas no son válidas
5	ingreso al sistema	en caso que las credenciales hayan sido validadas	cuando el usuario ingrese unas credenciales válidas	desplegar un mensaje de bienvenida, seguido del menú de opciones

Tabla 3.1 – Criterios de aceptación de historia de usuario US1

3.1.2. Diseño

3.1.2.1. Diagrama de Clases

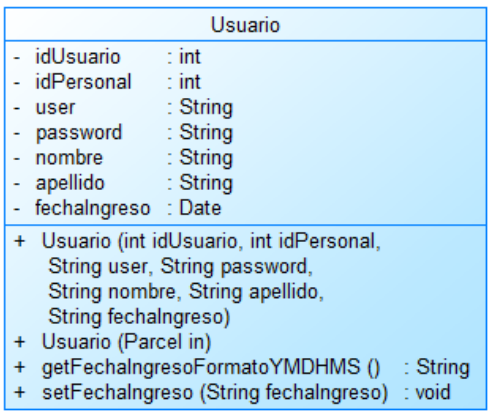


Ilustración 3.1 – Diagrama de clases en primera iteración

3.1.2.2. Diagrama Conceptual

USUARIO			
IDUSUARIO	<pi>	Long integer	<M>
IDPERFIL		Long integer	
USERUSUARIO		Variable characters (50)	
PASSWORDUSUARIO		Variable characters (50)	
NOMBREUSUARIO		Variable characters (50)	
APELLIDOUSUARIO		Variable characters (50)	
DIRECCIONUSUARIO		Variable characters (50)	
TELEFONOUSUARIO		Variable characters (50)	
EMAILUSUARIO		Variable characters (50)	
FECHAMODIFICACION		Date	
USUARIOMODIFICACION		Variable characters (50)	
idPersonal		Integer	
IDUSUARIO <pi>			

Ilustración 3.2 – Diagrama conceptual en primera iteración

3.1.2.3. Diagrama Conceptual – Aplicativo Móvil

Usuario			
IDUSUARIO	<pi>	Integer	<M>
idPersonal		Integer	
USERUSUARIO		Text	
PASSWORDUSUARIO		Text	
NOMBREUSUARIO		Text	
APELLIDOUSUARIO		Text	
FECHAINGRESO		Text	
IDUSUARIO <pi>			

Ilustración 3.3 – Diagrama conceptual del aplicativo móvil en primera iteración

3.1.2.4. Interfaz Gráfica

Ilustración 3.4 – Prototipo de interfaz gráfica de pantalla de ingreso

3.1.3. Codificación

3.1.3.1. Pruebas

Pruebas		
Número de Escenario	Aprobado	Tester
1	sí	Juan Esteban Canelos
2	sí	Juan Esteban Canelos
3	sí	Juan Esteban Canelos
4	sí	Juan Esteban Canelos
5	sí	Juan Esteban Canelos

Tabla 3.2 – Pruebas de historia de usuario US1

3.2. Segunda iteración

3.2.1. Análisis

Historias de usuario: US-2

Duración: 3 semanas

Criterios de aceptación:

Criterios de Aceptación				
Número de Escenario	Criterio de Aceptación	Contexto	Evento	Resultado / Comportamiento Esperado
1	guardar cuaderno de campo	en caso que exista una conexión a internet	cuando el usuario guarde el cuaderno de campo	el cuaderno de campo se guarda en la base de datos del museo
2	guardar cuaderno de campo localmente	en caso que no exista una conexión a internet	cuando el usuario guarde el cuaderno de campo	el cuaderno de campo se guarda en el celular hasta que pueda subirse a la base de datos del museo

Tabla 3.3 – Criterios de aceptación de historia de usuario US2

3.2.2. Diseño

3.2.2.1. Diagrama de Clases

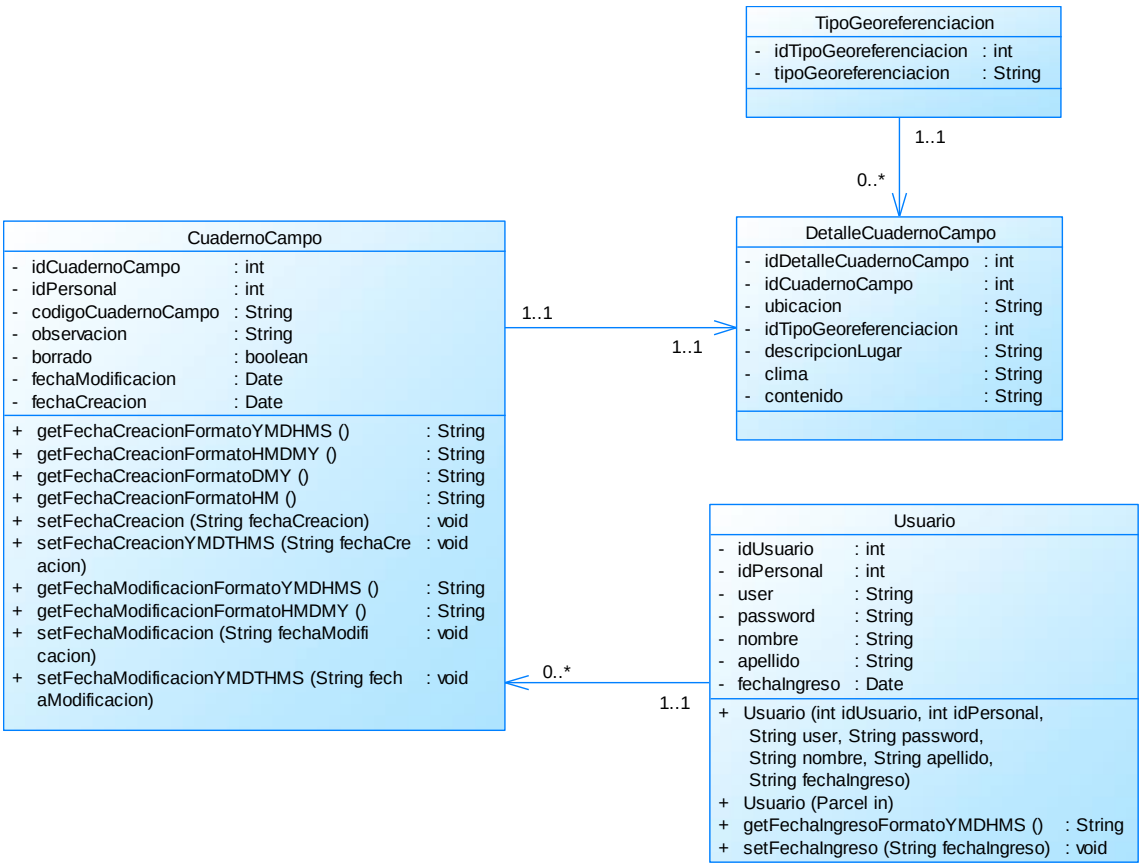


Ilustración 3.5 – Diagrama de clases en segunda iteración

3.2.2.2. Diagrama Conceptual

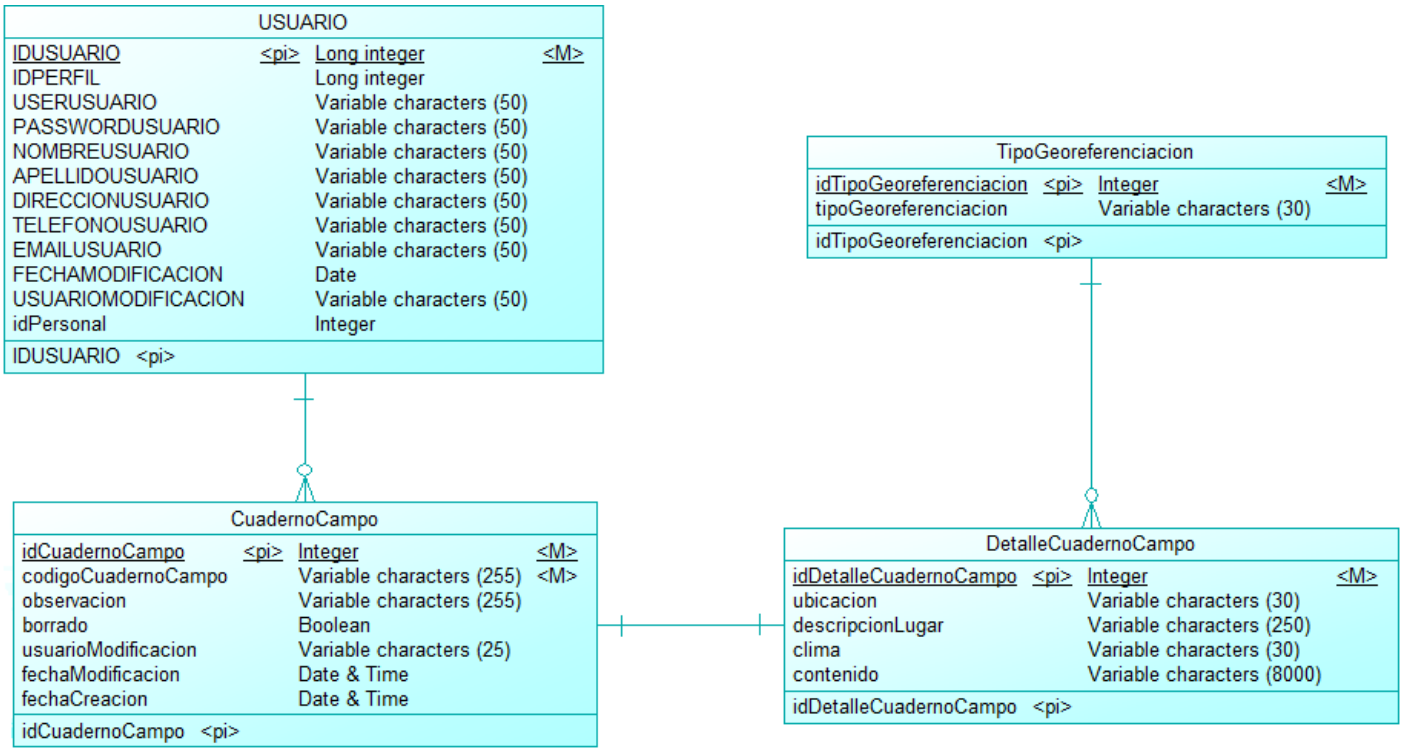


Ilustración 3.6 – Diagrama conceptual en segunda iteración

3.2.2.3. Diagrama Conceptual – Aplicativo Móvil

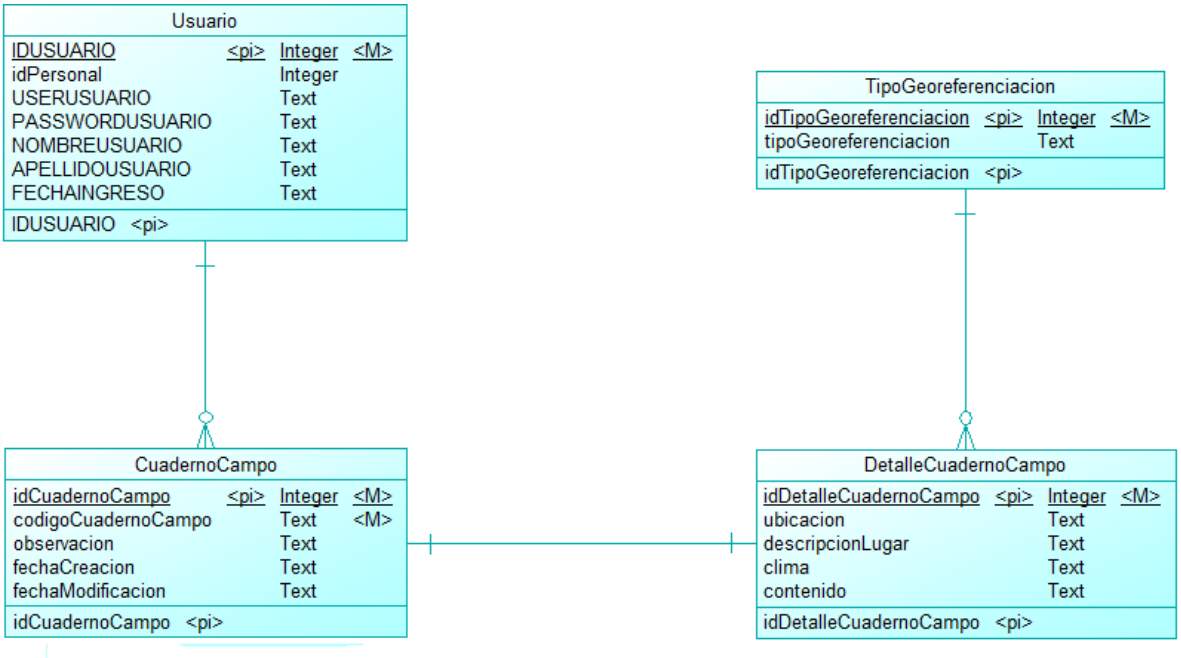


Ilustración 3.7 – Diagrama conceptual del aplicativo móvil en segunda iteración

3.2.2.4. Interfaz Gráfica

Ilustración 3.8 – Prototipo de interfaz gráfica de pantalla de ingreso de cuaderno de campo

Ilustración 3.9 – Prototipo de interfaz gráfica de pantalla de ingreso de cuaderno de campo

3.2.3. Codificación

3.2.3.1. Pruebas

Pruebas		
Número de Escenario	Aprobado	Tester
1	sí	Juan Esteban Canelos
2	sí	Juan Esteban Canelos

Tabla 3.4 – Pruebas de historia de usuario US2

3.3. Tercera iteración

3.3.1. Análisis

Historias de usuario: US-3

Duración: 2 semanas

Criterios de aceptación:

Criterios de Aceptación				
Número de Escenario	Criterio de Aceptación	Contexto	Evento	Resultado / Comportamiento Esperado
1	obtener ubicación	el servicio de ubicación se encuentra activo	cuando se solicite la ubicación	se registra la ubicación actual en el cuaderno de campo
2	no se encuentra activo el servicio de ubicación en el dispositivo	el servicio de ubicación no se encuentra activo	cuando se solicite la ubicación	se notifica al usuario que el servicio de ubicación se encuentra inactivo
3	obtener temperatura ambiental	el dispositivo cuenta con un sensor de temperatura ambiental	cuando se solicite la temperatura ambiental	se muestra la temperatura ambiental actual
4	no existe un sensor de temperatura ambiental en el dispositivo	el dispositivo no cuenta con un sensor de temperatura ambiental	cuando se solicite la temperatura ambiental	se notifica al usuario que el dispositivo no cuenta con un sensor de temperatura
5	obtener humedad ambiental	el dispositivo cuenta con un sensor de humedad ambiental	cuando se solicite la humedad ambiental	se muestra la humedad ambiental actual
6	no existe un sensor de humedad ambiental en el dispositivo	el dispositivo no cuenta con un sensor de humedad ambiental	cuando se solicite la humedad ambiental	se notifica al usuario que el dispositivo no cuenta con un sensor de humedad

Tabla 3.5 – Criterios de aceptación de historia de usuario US3

3.3.2. Diseño

3.3.2.1. Diagrama de Clases

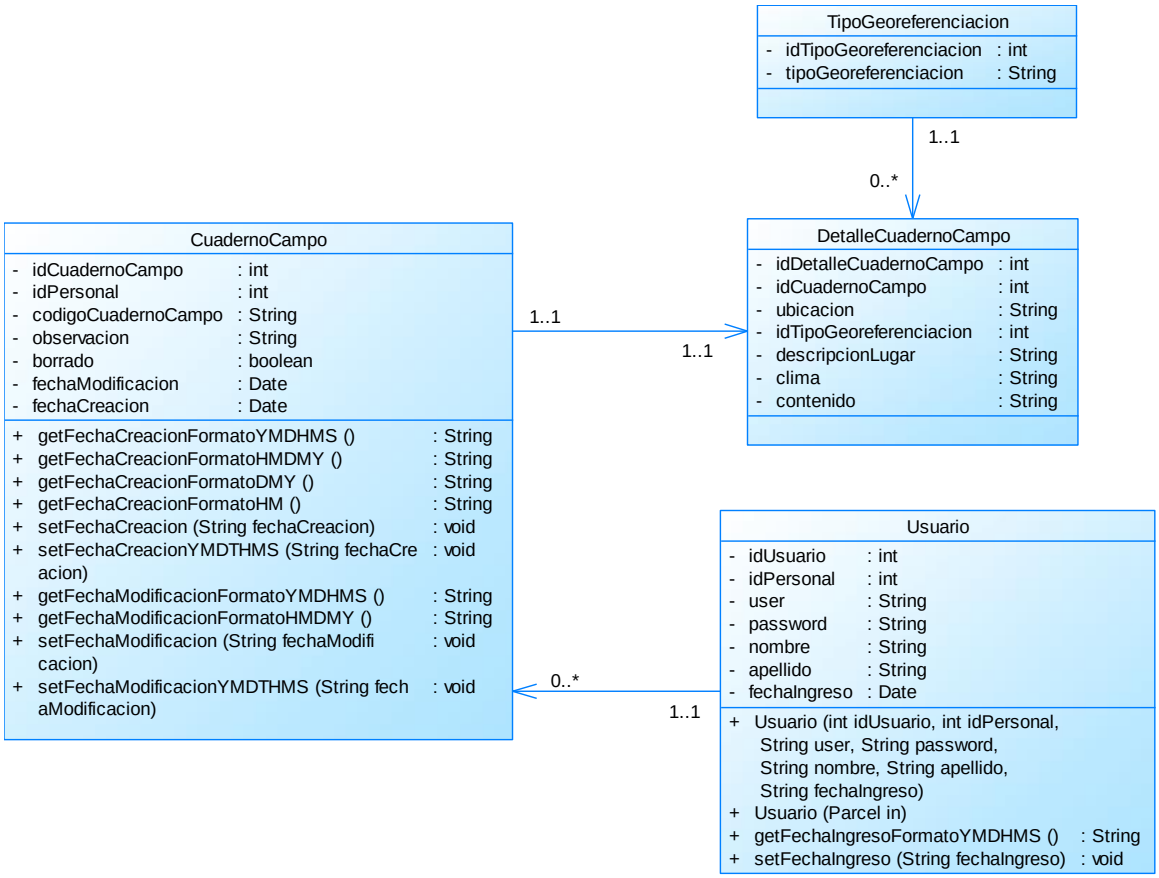


Ilustración 3.10 – Diagrama de clases en tercera iteración

3.3.2.2. Diagrama Conceptual

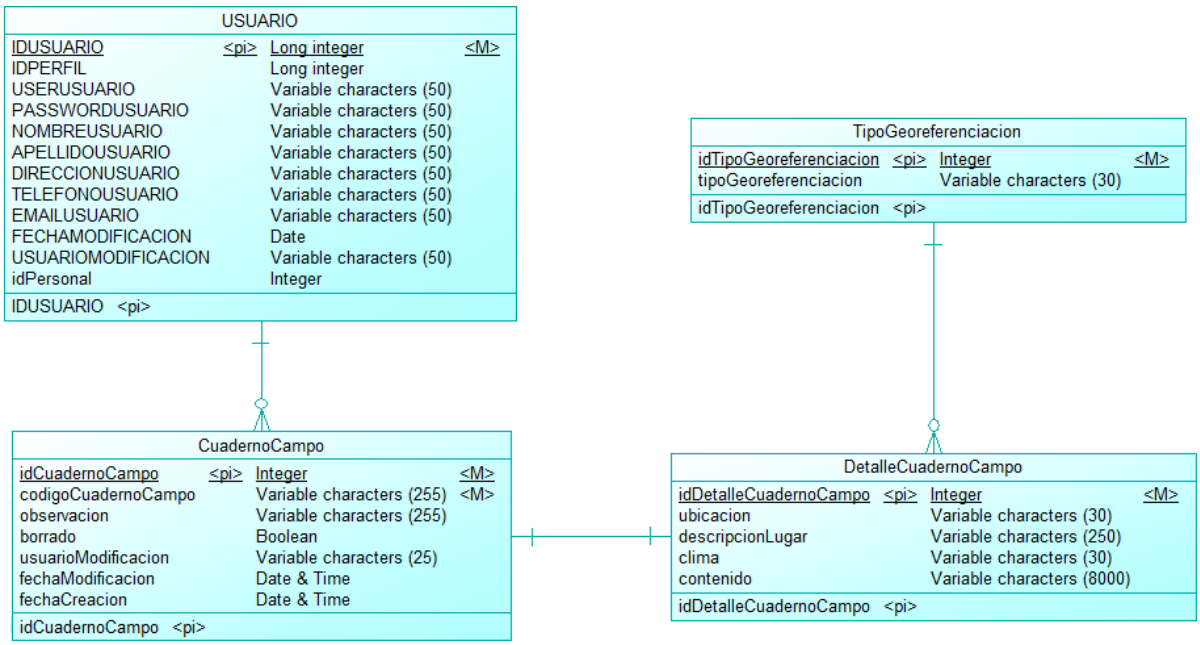


Ilustración 3.11 – Diagrama conceptual en tercera iteración

3.3.2.3. Diagrama Conceptual – Aplicativo Móvil

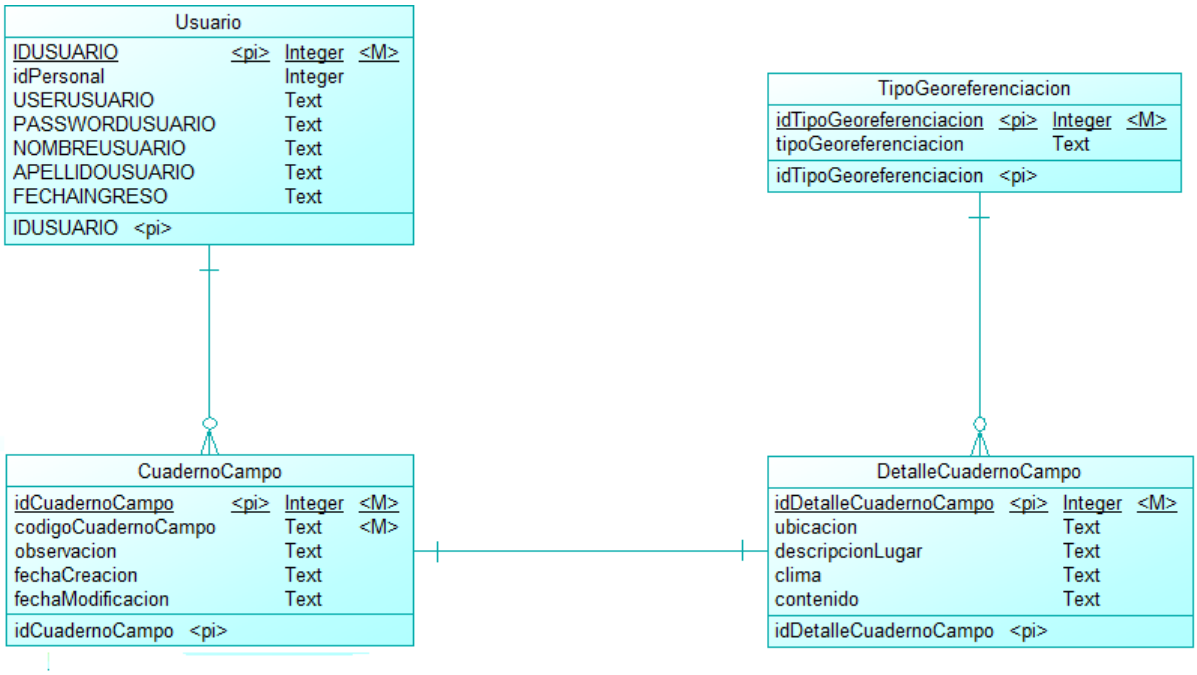


Ilustración 3.12 – Diagrama conceptual del aplicativo móvil en tercera iteración

3.3.2.4. Interfaz Gráfica

Ilustración 3.13 – Prototipo de interfaz gráfica de pantalla de uso de sensores

3.3.3. Codificación

3.3.3.1. Pruebas

Pruebas		
Número de Escenario	Aprobado	Tester
1	no	Juan Esteban Canelos
2	sí	Juan Esteban Canelos
3	sí	Juan Esteban Canelos
4	sí	Juan Esteban Canelos
5	sí	Juan Esteban Canelos
6	sí	Juan Esteban Canelos

Tabla 3.6 – Pruebas de historia de usuario US3

El comportamiento del escenario 1 durante las pruebas no fue el esperado, pues el sistema solo obtenía una ubicación aproximada, sin hacer uso del GPS para obtener la ubicación exacta del usuario. La historia de usuario no superó las pruebas de aceptación.

3.4. Cuarta iteración

3.4.1. Análisis

Historias de usuario: US-3, US-4

Duración: 2 semanas

Criterios de aceptación:

Criterios de Aceptación				
Número de Escenario	Criterio de Aceptación	Contexto	Evento	Resultado / Comportamiento Esperado
1	obtener ubicación	el servicio de ubicación se encuentra activo	cuando se solicite la ubicación	se registra la ubicación actual en el cuaderno de campo
2	no se encuentra activo el servicio de ubicación en el dispositivo	el servicio de ubicación no se encuentra activo	cuando se solicite la ubicación	se notifica al usuario que el servicio de ubicación se encuentra inactivo
3	obtener temperatura ambiental	el dispositivo cuenta con un sensor de temperatura ambiental	cuando se solicite la temperatura ambiental	se muestra la temperatura ambiental actual
4	no existe un sensor de temperatura ambiental en el dispositivo	el dispositivo no cuenta con un sensor de temperatura ambiental	cuando se solicite la temperatura ambiental	se notifica al usuario que el dispositivo no cuenta con un sensor de temperatura
5	obtener humedad ambiental	el dispositivo cuenta con un sensor de humedad ambiental	cuando se solicite la humedad ambiental	se muestra la humedad ambiental actual
6	no existe un sensor de humedad ambiental en el dispositivo	el dispositivo no cuenta con un sensor de humedad ambiental	cuando se solicite la humedad ambiental	se notifica al usuario que el dispositivo no cuenta con un sensor de humedad

Tabla 3.7 – Criterios de aceptación de historia de usuario US3

Criterios de Aceptación				
Número de Escenario	Criterio de Aceptación	Contexto	Evento	Resultado / Comportamiento Esperado
1	búsqueda de cuadernos por código	en caso que hayan resultados	cuando se realice la búsqueda	se desplegará un listado con el código, fecha de creación y fecha de modificación de los resultados, mostrando 10 cuadernos por página
2	búsqueda de cuadernos por código sin resultados	en caso que no hayan resultados	cuando se realice la búsqueda	se desplegará un mensaje que indicia que no se encontraron cuadernos de campo
3	búsqueda de cuadernos por código sin resultados	en caso que no hayan resultados	cuando se realice la búsqueda	se desplegará un mensaje que indicia que no se encontraron cuadernos de campo
4	búsqueda de cuadernos en un rango de fechas	en caso que hayan resultados	cuando se realice la búsqueda	se desplegará un listado con el código, fecha de creación y fecha de modificación de los resultados
5	búsqueda de cuadernos en un rango de fechas sin resultados	en caso que no hayan resultados	cuando se realice la búsqueda	se desplegará un mensaje que indica que no se encontraron cuadernos de campo
6	búsqueda sin conexión	en caso que no haya conexión a internet	cuando se realice la búsqueda	se desplegará un mensaje que indica que no existe conexión

Tabla 3.8 – Criterios de aceptación de historia de usuario US4

3.4.2. Diseño

3.4.2.1. Diagrama de Clases

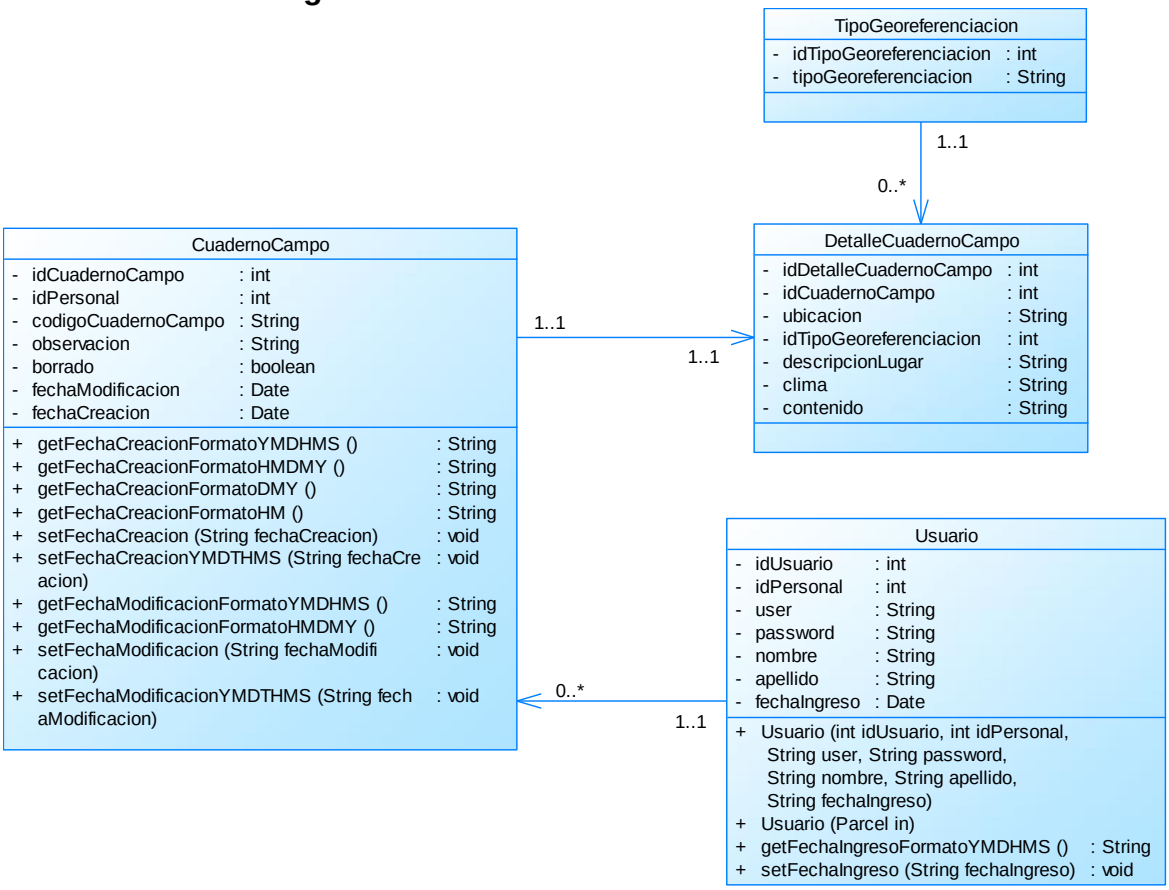


Ilustración 3.14 – Diagrama de clases en cuarta iteración

3.4.2.2. Diagrama Conceptual

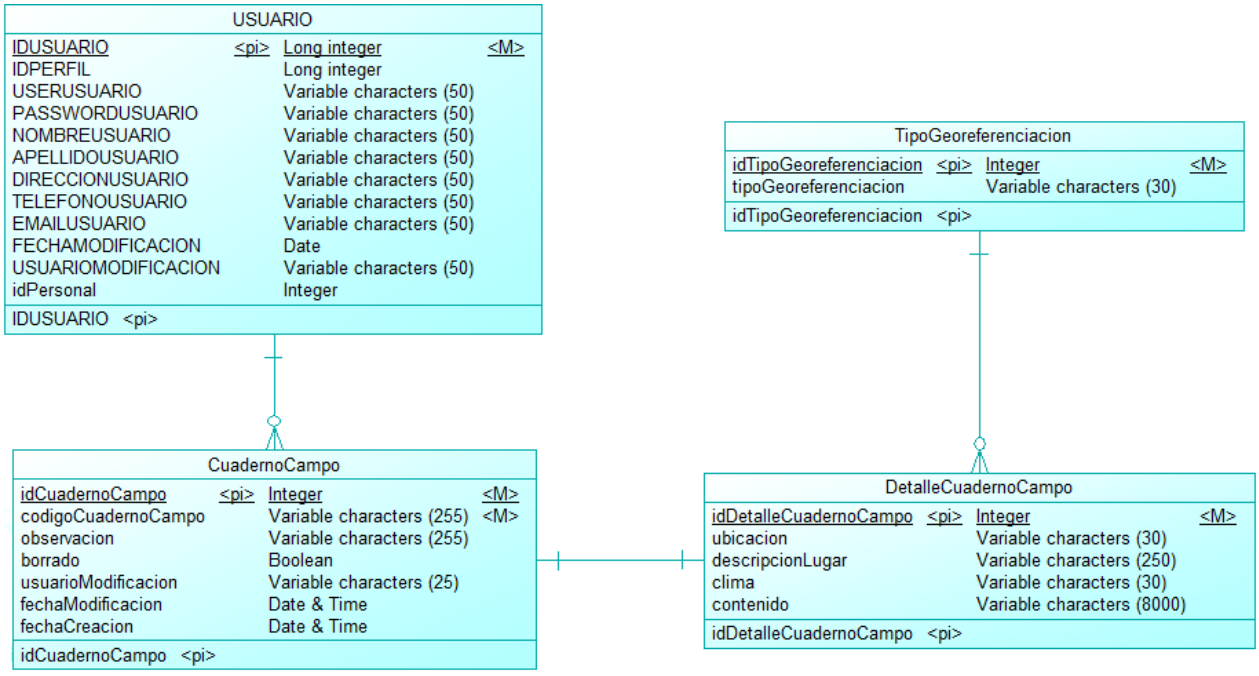


Ilustración 3.15 – Diagrama conceptual en cuarta iteración

3.4.2.3. Diagrama Conceptual – Aplicativo Móvil

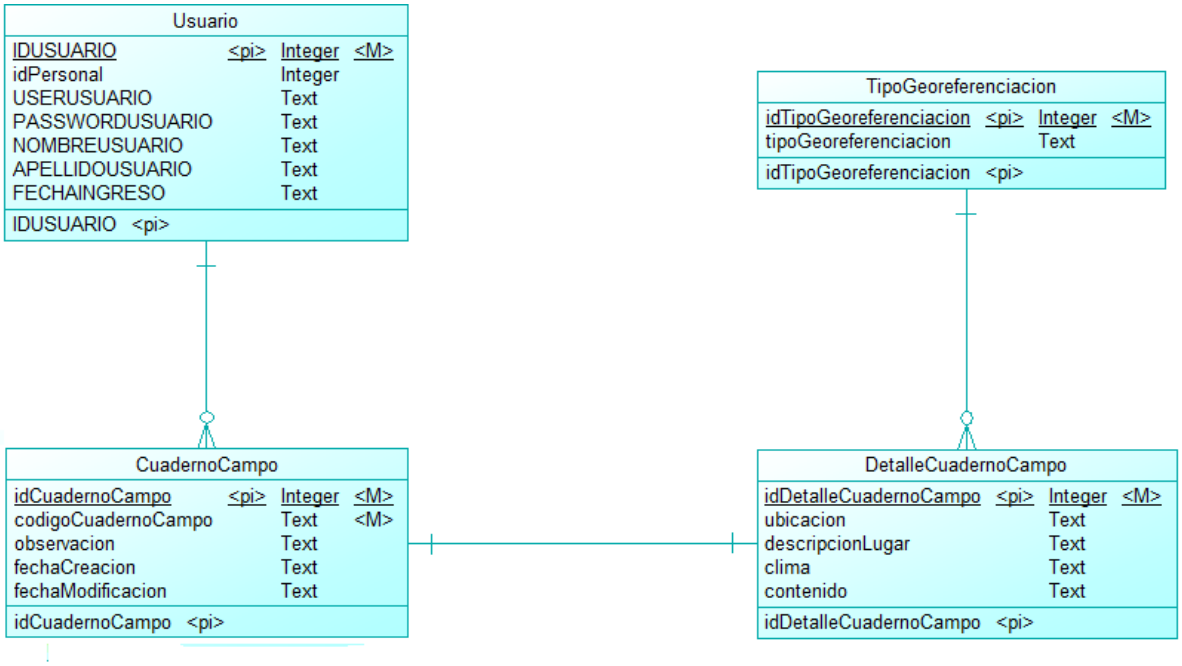


Ilustración 3.16 – Diagrama conceptual del aplicativo móvil en cuarta iteración

3.4.2.4. Interfaz de Usuario

Ilustración 3.17 – Prototipo de interfaz gráfica de pantalla de búsqueda de cuadernos de campo

3.4.3. Codificación

3.4.3.1. Pruebas

Pruebas		
Número de Escenario	Aprobado	Tester
1	sí	Juan Esteban Canelos
2	sí	Juan Esteban Canelos
3	sí	Juan Esteban Canelos
4	sí	Juan Esteban Canelos
5	sí	Juan Esteban Canelos
6	sí	Juan Esteban Canelos

Tabla 3.9 – Pruebas de historia de usuario US3

Pruebas		
Número de Escenario	Aprobado	Tester
1	sí	Juan Esteban Canelos
2	sí	Juan Esteban Canelos
3	sí	Juan Esteban Canelos
4	sí	Juan Esteban Canelos
5	sí	Juan Esteban Canelos
6	sí	Juan Esteban Canelos

Tabla 3.10 – Pruebas de historia de usuario US4

Las tareas no aprobadas en la iteración anterior fueron completadas, junto con una nueva historia de usuario.

3.5. Quinta iteración

3.5.1. Análisis

Historias de usuario: US-5

Duración: 3 semanas

Criterios de aceptación:

Criterios de Aceptación				
Número de Escenario	Criterio de Aceptación	Contexto	Evento	Resultado / Comportamiento Esperado
1	editar un cuaderno de campo	seleccionar un cuaderno de la lista de resultados	cuando se seleccione un cuaderno de campo	se desplegará un menú que permitirá su edición
2	editar información de un cuaderno de campo	en caso que exista una conexión a internet	cuando se intente guardar la información actualizada del cuaderno de campo	se mostrará un mensaje que indique que el cuaderno de campo se ha actualizado correctamente
3	editar información de un cuaderno de campo	en caso que no exista una conexión a internet	cuando se intente guardar la información actualizada del cuaderno de campo	se mostrará un mensaje que indique que el cuaderno de campo no ha podido ser actualizado
4	eliminar un cuaderno de campo	en caso que exista una conexión a internet	cuando se intente eliminar el cuaderno de campo	se mostrará un mensaje que indique que el cuaderno de campo se ha eliminado correctamente
5	eliminar un cuaderno de campo	en caso que no exista una conexión a internet	cuando se intente eliminar el cuaderno de campo	se mostrará un mensaje que indique que el cuaderno de campo no ha podido ser eliminado

Tabla 3.11 – Criterios de aceptación de historia de usuario US5

3.5.2. Diseño

3.5.2.1. Diagrama de Clases

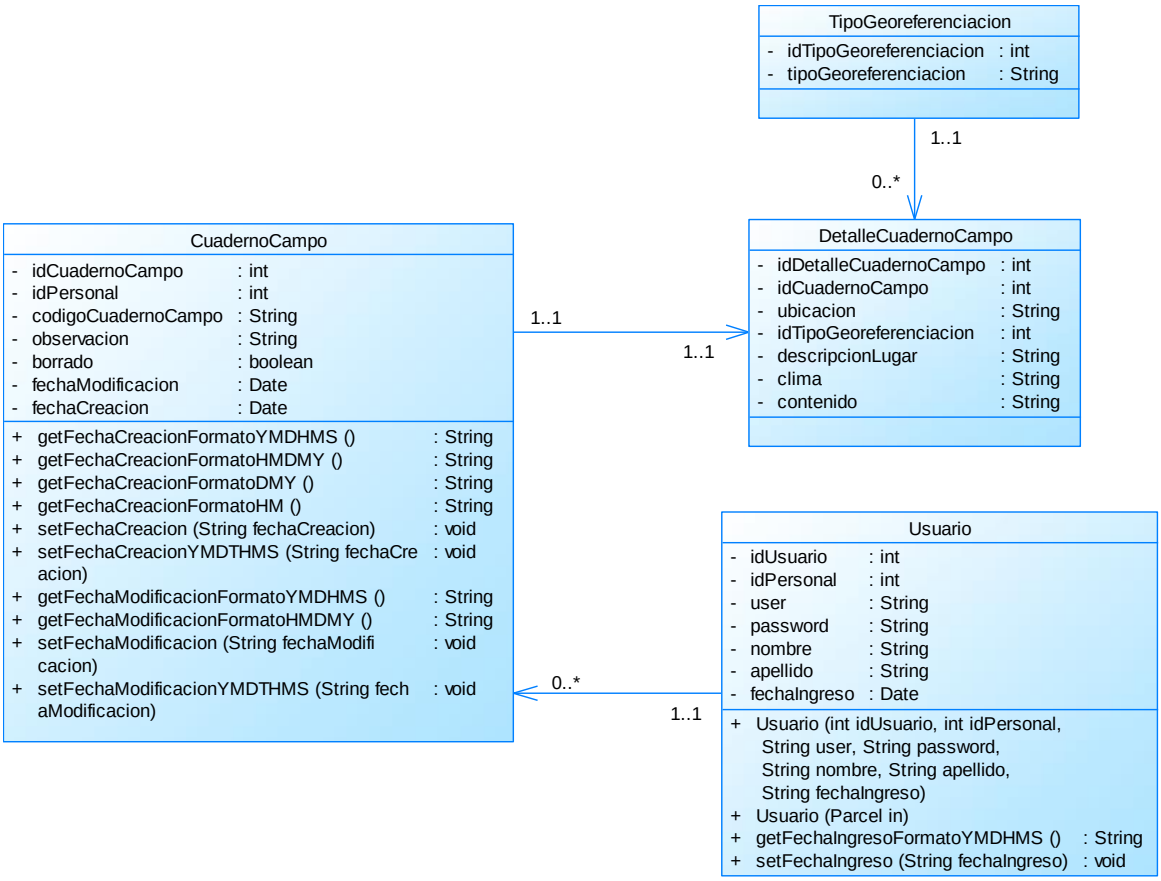


Ilustración 3.18 – Diagrama de clases en quinta iteración

3.5.2.2. Diagrama Conceptual

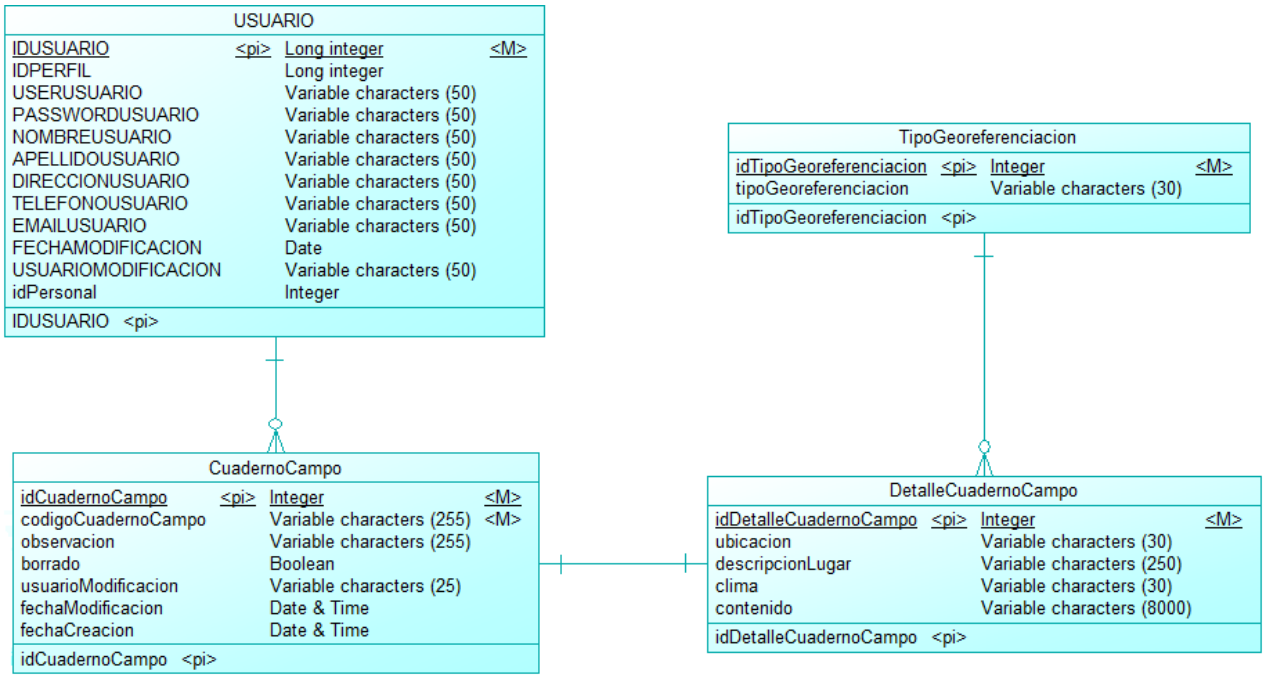


Ilustración 3.19 – Diagrama conceptual en quinta iteración

3.5.2.3. Diagrama Conceptual – Aplicativo Móvil

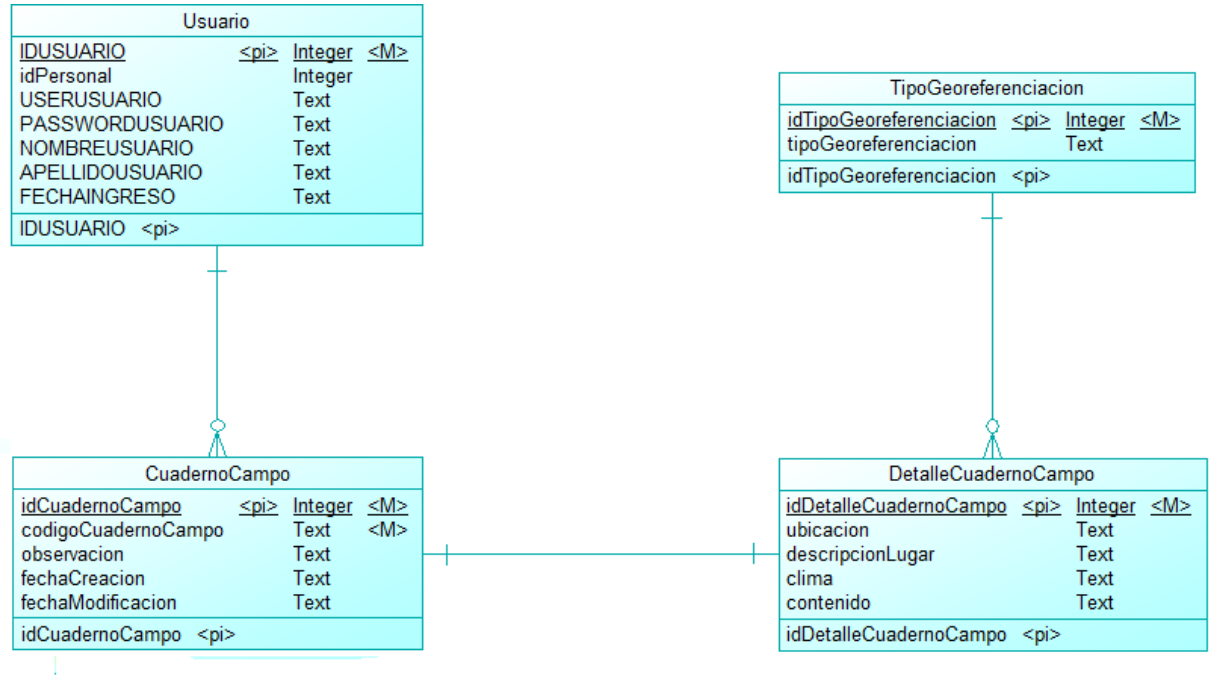


Ilustración 3.20 – Diagrama conceptual del aplicativo móvil en quinta iteración

3.5.2.4. Interfaz de Usuario

Ilustración 3.21 – Prototipo de interfaz gráfica de pantalla de modificación y eliminación de cuadernos de campo

3.5.3. Codificación

3.5.3.1. Pruebas

Pruebas		
Número de Escenario	Aprobado	Tester
1	sí	Juan Esteban Canelos
2	sí	Juan Esteban Canelos
3	sí	Juan Esteban Canelos
4	sí	Juan Esteban Canelos
5	sí	Juan Esteban Canelos

Tabla 3.12 – Pruebas de historia de usuario US5

4. Implementación

4.1. Diagrama de Despliegue

El sistema está dividido en 3 bloques:

- el dispositivo móvil del usuario, donde se encuentra el aplicativo móvil
- el servidor de aplicaciones, que engloba los servicios web REST expuestos, y las diferentes capas subyacentes del sistema
- la base de datos del museo QCAZ

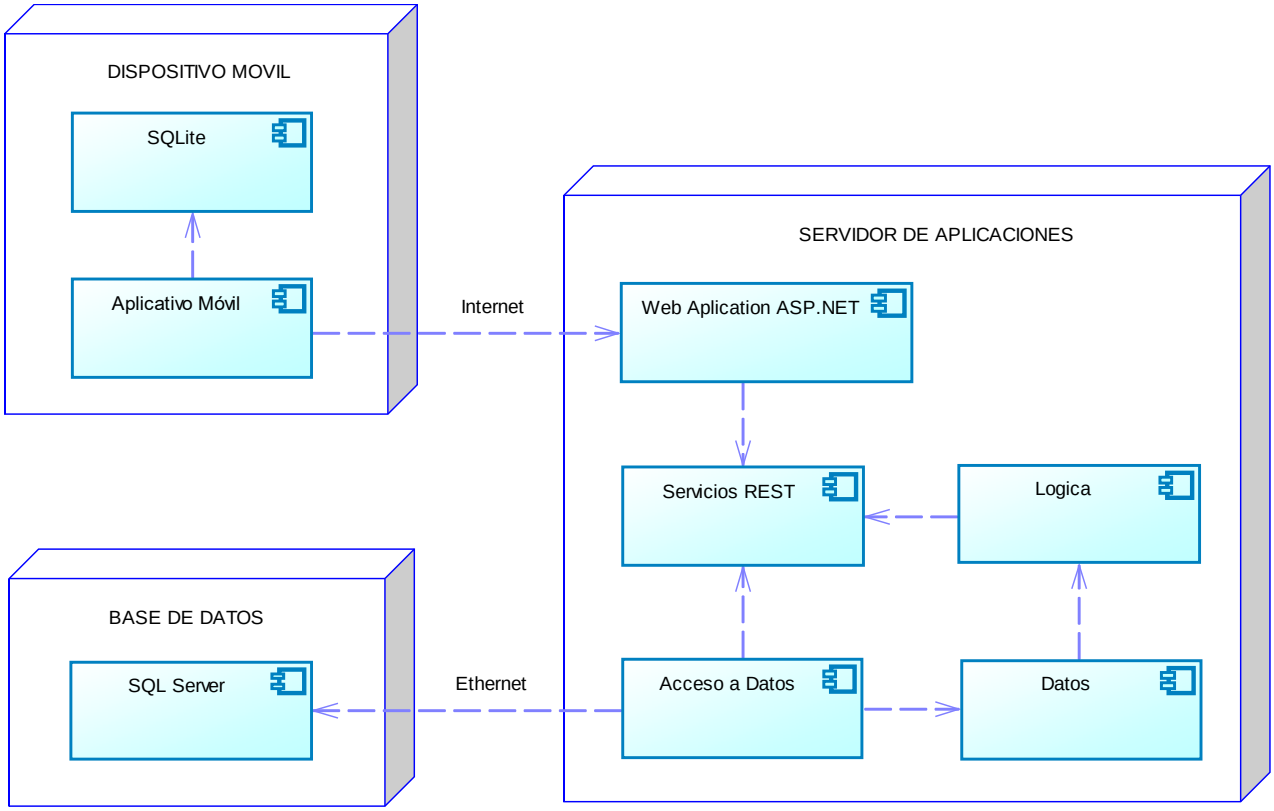


Ilustración 4.1 – Diagrama de despliegue

4.2. Instalación y Configuración

Para instalar el aplicativo móvil, basta con obtener el instalador (archivo .apk) del personal de Biología, y correrlo en el dispositivo deseado. Una vez finalizada la instalación, el aplicativo está listo para usar.

4.3. Manual de Usuario

(Ver manual de usuario – anexo 1)

CONCLUSIONES

- La increíble presencia y el gran alcance de la tecnología en estos tiempos es algo que puede aprovecharse en cualquier rama de la ciencia, como fue en este caso en la Biología. Después de analizar la situación actual en el capítulo 2, se pudo ver que los dispositivos móviles y el uso de aplicaciones tecnológicas pueden lograr automatizar y optimizar tareas, y permitir un mejor manejo de la información y una optimización de los recursos disponibles.
- Después de investigar sobre los servicios web en el capítulo 1, e implementarlos durante el capítulo 2, se pudo observar que los servicios web proporcionan una comunicación interoperable entre los diferentes sistemas utilizados, independientemente de su fabricante o de su tecnología. Los servicios permiten una fácil escalabilidad en caso de necesitar la implementación de nuevas características para el sistema desarrollado, o incluso si se quisieran usar las mismas funciones en otro sistema. Si los servicios web desarrollados requirieran un alto nivel de seguridad, desarrollarlos con SOAP hubiera sido una mejor alternativa.
- El desarrollo del proyecto durante el capítulo 3, mediante la metodología eXtreme Programming, permitió un desarrollo eficiente y sin retrasos, pues al mantener un contacto constante con el cliente, las correcciones y los cambios en la aplicación móvil pueden realizarse de manera rápida, y sin provocar muchos inconvenientes. Esto permitió cumplir con las necesidades del cliente en el plazo establecido.

RECOMENDACIONES

- Debido a que las historias de usuario son la base de todo el desarrollo en la metodología eXtreme Programming, se recomienda que estas sean revisadas y verificadas en su totalidad con el cliente, de preferencia de manera escrita, antes de iniciar el proyecto, como se realizó en el capítulo 2 en la toma de requerimientos.
- Las pruebas son una parte indispensable en la metodología eXtreme Programming, que deben realizarse y repetirse en cada iteración, de manera que se asegure el correcto funcionamiento de los entregables anteriores. Si bien existen algunos programas y librerías que permiten realizar pruebas automatizadas, se recomienda evaluar el tiempo y esfuerzo que demanda su programación en relación al tiempo y esfuerzo de llevar a cabo las pruebas manuales, ya que la manera más óptima puede variar según la complejidad del proyecto. Durante el desarrollo de este aplicativo móvil, en el capítulo 3, se observó que las pruebas realizadas de manera manual exigían poco esfuerzo, por lo que se optó por realizar las pruebas de esta manera.
- En el capítulo 3, durante el diseño de los prototipos de las interfaces de usuario, se utilizó un programa que ilustraba el dispositivo móvil y sus componentes reales, sin la necesidad de codificar. Se recomienda utilizar este tipo de programas informáticos para el diseño, ya que permiten visualizar de mejor manera el resultado que se obtendrá en el desarrollo, evitando cambios posteriores en el diseño después de que se haya implementado. Esto también permitirá visualizar las interfaces de usuario en dispositivos de diferentes tamaños de manera sencilla.
- Como se pudo observar durante el desarrollo del proyecto en el capítulo 3, se recomienda utilizar la metodología eXtreme Programming para realizar proyectos donde el desarrollo debe ser rápido, y pueda mantenerse una comunicación alta y constante con el cliente, evitando de esta manera retrasos grandes por cambios en los requerimientos. Esta metodología también es ideal para grupos de desarrollo de pocas personas o incluso de una sola persona, ya que presenta la ventana de ser fácilmente escalable.

BIBLIOGRAFÍA

- Abellán Martínez, M. (2015). *Universidad de Murcia.* Obtenido de Tutorial Java Web: http://dis.um.es/~lopezquesada/documentos/IES_1516/IAW/curso/UT3/ActividadesAlumnos/grupo9/index.html
- Android Studio. (s.f.). *Android Studio*. Obtenido de Cómo agregar código C y C++ a tu proyecto: https://developer.android.com/studio/projects/add-native-code.html
- Benítez, M. Á., & Arias, Á. (2017). *Curso de Introducción a la Administración de Bases de Datos.* IT Campus Academy.
- CMM. (2017). *CMM*. Obtenido de Lenguajes de Programación: http://es.ccm.net/contents/304-lenguajes-de-programacion
- CMS. (17 de Febrero de 2005). *Centers for Medicare & Medicaid Services.* Obtenido de Selecting a Development Approach: https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf
- De Seta, L. (12 de Abril de 2010). *Dos Ideas*. Obtenido de James Gosling, el padre de Java, renuncia a Oracle: https://dosideas.com/noticias/java/880-james-gosling-el-padre-de-java-renuncia-a-oracle
- *Desde Linux*. (24 de Mayo de 2016). Obtenido de Carácteristicas y cualidades de Android Studio: https://blog.desdelinux.net/caracteristicas-y-cualidades-de-android-studio/
- El País. (20 de Abril de 2009). *El País*. Obtenido de Oracle adquiere Sun Microsystems por 5.710 millones: https://elpais.com/tecnologia/2009/04/20/actualidad/1240216080_850215.html
- García Valcárcel, I., & Munilla Calvo, E. (2003). *E-Business Corporativo: Cómo Implantar Software Libre, Servicios Web y el Grid Computing para Ahorrar Costes y Mejorar las Comunicaciones en su Empresa.* Madrid: FC Editorial.
- García, F. (25 de Enero de 2013). *Wordpress*. Obtenido de Entorno de Desarrollo Integrado (IDE): https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/
- IEEE. (1990). *610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology.* Nueva York.
- IEEE. (1997). *1074-1997 - IEEE Standard for Developing Software Life Cycle Processes.* Nueva York.
- IEEE. (1998). *830-1998 - IEEE Recommended Practice for Software Requirements Specifications.* Nueva York.
- ISO. (2008). *ISO/IEC 12207:2008.*

- Lamarca Puente, M. J. (8 de Diciembre de 2013). *Hipertexto*. Obtenido de Servicios Web: http://www.hipertexto.info/documentos/serv_web.htm
- McGoogan , C. (10 de Febrero de 2016). *The Telegraph*. Obtenido de App revenue will overtake the music industry this year: http://www.telegraph.co.uk/technology/2016/02/10/app-revenue-will-overtake-the-music-industry-this-year/
- Microsoft. (3 de Abril de 2017). *Microsoft*. Obtenido de C#: https://docs.microsoft.com/es-es/dotnet/csharp/csharp
- Microsoft. (s.f.). *Microsoft*. Obtenido de ADO.NET Entity Framework: https://msdn.microsoft.com/es-es/library/bb399572(v=vs.100).aspx
- Microsoft. (s.f.). *Microsoft*. Obtenido de .NET Framework: https://www.microsoft.com/net/download/framework
- Microsoft. (s.f.). *Microsoft*. Obtenido de Compare Visual Studio 2017 IDEs: https://www.visualstudio.com/vs/compare/
- Mullis, A. (30 de Mayo de 2017). *Android Authority*. Obtenido de Android Studio tutorial for beginners: http://www.androidauthority.com/android-studio-tutorial-beginners-637572/
- Nevado Cabello, M. V. (s.f.). *Introducción a las Bases de Datos Relacionales.* Madrid: Vision Libros.
- Pérez Valdés, D. (26 de Octubre de 2007). *Maestros del Web*. Obtenido de ¿Qué son las bases de datos?: http://www.maestrosdelweb.com/que-son-las-bases-de-datos/
- Pressman, R. S. (2010). *Ingeniería de Software: Un enfoque práctico.* México: McGrawHill.
- R., A. (29 de Mayo de 2014). *Hipertextual*. Obtenido de XML, JSON, YAML: Formatos para intercambiar información: https://hipertextual.com/archivo/2014/05/xml-json-yaml/
- Real Academia Española. (2001). *Diccionario de la Lengua Española.* Madrid: Espasa.
- Roa, P. A., & Vargas, C. (2009). El Cuaderno de Campo como Estrategia de Enseñanza en el Departamento de Biología de la UPN. *Bio-Grafía*, 61-73.
- Sánchez Tapia, J. C. (18 de Abril de 2013). *jc*. Obtenido de RESTful Web Services: http://jc.pe/2013/04/18/restful-web-services/
- Stringfellow, A. (31 de Marzo de 2017). *Stackify*. Obtenido de SOAP vs. REST: The Differences and Benefits Between the Two Widely-Used Web Service Communication Protocols: https://stackify.com/soap-vs-rest/

- Tutorials Point. (s.f.). *Tutorials Point*. Obtenido de Java - Overview: https://www.tutorialspoint.com/java/java_overview.htm
- W3C. (27 de Abril de 2007). *W3C*. Obtenido de SOAP Version 1.2 Part 1: Messaging Framework (Second Edition): https://www.w3.org/TR/soap12-part1/
- Wells, D. (8 de Octubre de 2013). *Extreme Programming.* Obtenido de Extreme Programming: A gentle introduction: http://www.extremeprogramming.org/