

Arquitectura, diseño, y configuración de un simulador de bicis compartidas

Carlos Ruiz Ballesteros

02 de Agosto de 2018

Índice

Resumen	3
1. Introducción	5
1.1. Motivación	5
1.2. Contexto	6
1.3. Objetivos	7
1.4. Metodología	9
2. Test	11
Referencias	12

Resumen

Imaginemos por un momento que necesitamos ir de un sitio a otro en nuestra ciudad y disponemos de un sistema de bicicletas por estaciones. Como sabemos los sistemas de bicicletas públicas ofrecen a los usuarios bicis en estaciones situadas en puntos concretos de la ciudad para poder utilizarlas como medio de transporte de un punto a otro. Estas estaciones están situadas en lugares concretos de la ciudad y podemos desplazarnos de un sitio a otro, cogiendo las bicis en una estación y dejándolas en otra cerca de nuestro destino.

Con la llegada de los smartphones, la utilización de estos sistemas es más sencilla, ya que los usuarios pueden disponer de mucha información del sistema a través de aplicaciones software instaladas en estos dispositivos. Estas aplicaciones pueden ofrecerle al usuario estaciones donde coger y dejar una bici en función del destino al que quieran ir. En sistemas como BiciMad es posible reservar bicicletas, ver el estado actual de todas las estaciones, etc.

En ciertas ocasiones, este sistema puede saturarse en estaciones concretas, haciendo que la experiencia de los usuarios que quieran utilizarlo empeore, por lo que es necesario aplicar soluciones. Hay que plantear posibles estrategias de balanceo de prever esas situaciones problemáticas. El objetivo es balancear las estaciones para que las estas puedan ser utilizadas para coger o devolver una bici el mayor tiempo que sea posible, minimizando los costes.

Muchas soluciones y estrategias de balanceo son muy difíciles de probar directamente en el sistema, ya que habría que implementar muchas cosas en un sistema en funcionamiento, y quizás nuestra estrategia de balanceo no sea la más adecuada, ocasionando gastos innecesarios y empeorando la experiencia de los usuarios. Para probar la viabilidad de las estrategias que se planteen es necesario disponer de un **simulador** capaz de probar estos sistema y algoritmos de balanceo propuestos.

En eso se centra este trabajo, en la implementación de un simulador capaz de recrear de la manera más realista posible, un sistema de bicis compartidas. Este simulador deberá permitir la posibilidad de tener diferentes tipos de usuario, implementar un sistema de recomendaciones, implementar un sistema de incentivos y poder probar cualquier sistema de bicis públicas en cualquier lugar del mundo. En definitiva, crear un simulador que nos permita crear, simular y analizar situaciones reales e implementar sistemas de balanceo para comprobar su viabilidad. Se crearán archivos de configuración para crear situaciones con estaciones y ciudades reales. Posteriormente tras las simulaciones se podrán analizar los datos obtenidos y visualizarlos para comprobar su eficacia.

1. Introducción

En este capítulo se darán a conocer las motivaciones que llevaron a la realización de este proyecto, el contexto del mismo y los objetivos planteados.

1.1. Motivación

El proyecto consiste en la realización de un simulador de sistemas de bicis compartidas en entornos urbanos, como por ejemplo BiciMad¹ en Madrid, o Vèlib² en Francia.

La necesidad de alentar a las personas a utilizar vías alternativas de transporte a las comúnmente usadas es cada vez más necesario. El aumento de la población en las grandes ciudades, el aumento de CO2 son hechos que nos obligan a pensar como cambiar y/o mejorar nuestros medios de transporte público.

Los sistemas de bicis compartidas instaladas en las grandes ciudades son una muy buena opción a la hora de buscar alternativas de movilidad. Estos sistemas permiten a los ciudadanos moverse por distintos puntos de la ciudad alquilando una bici de cualquiera de las estaciones disponibles y devolviéndola en otra estación diferente.

Pero el problema no solo se centra en promover el uso de estos sistemas, sino que va más allá. Uno de los más importantes es el balanceo de bicicletas entre estaciones. Algunas situaciones que pueden producir este desequilibrio son:

- Eventos especiales.
- Horas punta.
- Aglomeraciones inesperadas.

Para evitar estos problemas es necesario tener un control sobre los recursos disponibles, manejarlos y distribuirlos de la manera más eficiente posible. Llevar a cabo nuevas ideas en un sistema de alquiler de bicis y probarlo en la realidad, puede ser muy tedioso y costoso.

Es por esto por lo que surge la necesidad de un simulador, con el que podamos poner a prueba todos estos tipos de sucesos, crear distintos tipos de usuarios, probar sistemas de recomendaciones o de incentivos, para que los usuarios puedan ayudar a balancear el sistema.

¹BiciMad: <https://bicimad.com/>.

²Vèlib: <https://www.velib-metropole.fr/>.

Un simulador es útil para el estudio de estos problemas donde se pueda probar en cualquier ciudad del mundo distintos sistemas, algoritmos e ideas posibles de implementar para incentivar su uso, mejorar el sistema, o incluso probar soluciones en algunas ciudades y trasladarlas a otras.

Por otro lado, no solo está la posibilidad de utilizar el simulador como una herramienta para sacar conclusiones respecto al funcionamiento del sistema en sí, sino que además cabe la posibilidad de analizar el comportamiento de diferentes tipos de usuario en este tipo de sistemas. Para ello es necesario que se pueda añadir a nivel de código usuarios para que cualquier persona con conocimientos de programación pueda implementar sus propios algoritmos de comportamiento del simulador.

Adicionalmente, no debería ser un simulador único e invariable, sino que debe ser posible su modificación y adaptación a las necesidades de cada investigación, pero dentro del ámbito de los sistemas de bicis compartidas.

1.2. Contexto

Para definir las diferentes partes de nuestro simulador, es necesario tener una visión general del sistema de bicis en la realidad.

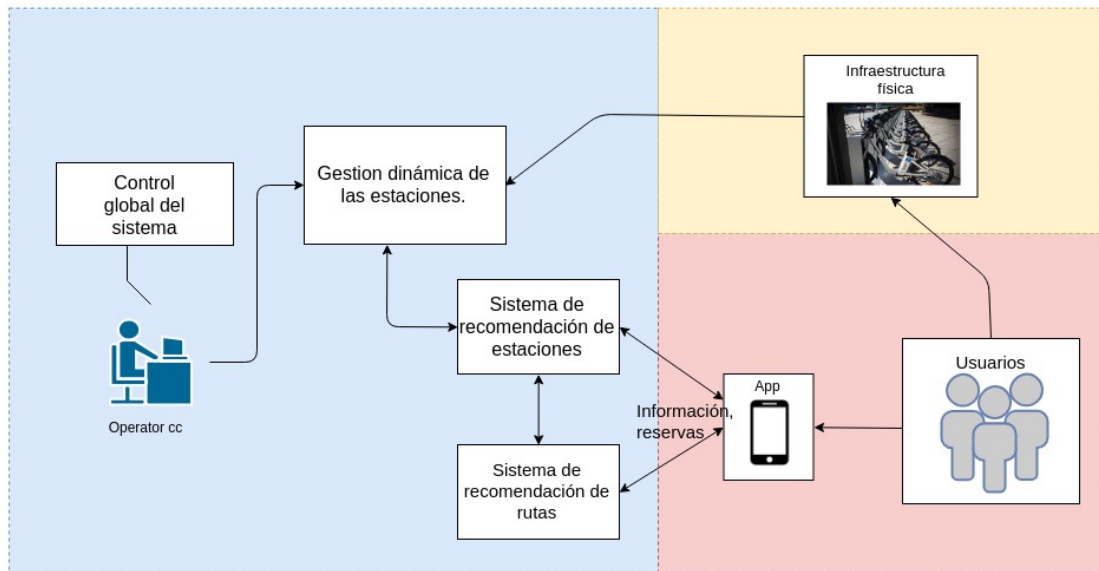


Figura 1: Sistema de bicis compartidas

Podemos distinguir claramente en la Figura 1 tres partes principales dentro del sistema de bicis compartidas:

- Infraestructura física con estaciones y bicis (Amarillo).
- Usuarios con Smartphone y App del sistema (Rojo).
- Sistema de recomendaciones y gestión del sistema (Azul).

Estas tres partes diferenciadas constituyen tres partes importantes dentro del desarrollo del simulador. El **núcleo**, que incluirá la infraestructura y como deben los usuarios interactuar con el sistema, las implementaciones de los **usuarios** que implementarán distintas formas de actuar dentro del sistema, y el **sistema de recomendaciones** que podrá influir en el comportamiento de los usuarios.

Los usuarios hacen uso de la infraestructura cuando cogen o dejan una bici y también hacen uso del sistema de recomendaciones a través de la App, para reservar una bici o un hueco, ver el estado de una estación, o quizás hacer caso a alguna de las sugerencias de la App.

Vemos que hay una interacción continua entre usuarios e infraestructura. El sistema de recomendaciones puede influenciar en las decisiones finales del usuario.

En el desarrollo de este simulador hemos participado varias personas hasta la fecha de publicación de esta memoria, las cuales han realizado diferentes partes, aunque parte de este desarrollo ha sido realizado de forma conjunta debido a la necesidad de tener una base común, que serían el núcleo y algunos estándares definidos.

Con esta visión global del sistema se especifica una serie de objetivos que se detallan a continuación.

1.3. Objetivos

Antes de plantear que objetivos perseguir, hay que analizar que necesidades va a tener el simulador en el futuro y con que fin se va a utilizar en términos globales. Podríamos distinguir varios objetivos si hablamos de la totalidad del simulador:

- Probar algoritmos de balanceo y predicción de demanda en un sistema de bicicletas compartido.
- Implementar sistemas de recomendaciones e incentivos para optimizar la demanda.
- Plantear nuevas distribuciones de estaciones sobre una ciudad.

Basado en estos objetivos globales se derivan los siguientes objetivos más concretos para el desarrollo:

- Recrear infraestructuras reales en ciudades reales.
- Generar usuarios en cualquier punto de la ciudad.
- Generación de usuarios versátil y que puedan seguir distribuciones (Poisson).
- Poder definir qué tipos de usuarios queremos en nuestro sistema y parametrizarlos para que puedan tener comportamientos variados.
- Poder definir qué tipos de usuarios queremos en nuestro sistema y parametrizarlos para que puedan tener comportamientos variados.
- Facilidad para crear configuraciones (GUI).
- Creación de distintos tipos de usuario, con diferentes comportamientos que respondan de forma distinta a las situaciones dadas y las recomendaciones.
- Simulación realista.
- Análisis de los datos para probar los diferentes algoritmos.

En general el simulador tiene que ser capaz de recrear situaciones reales basadas en entornos reales, con infraestructuras existentes o que puedan existir en el mundo real. Es decir, uno de los objetivos básicos es dotar al simulador de mecanismos ágiles para poder crear configuraciones de diferentes situaciones.

Además estas configuraciones tienen que poder ser generadas con cierta independencia del simulador, añadiendo la posibilidad de que nuevos desarrolladores puedan crear sus propias herramientas que generen casos para la simulación y ofreciendo una interfaz agradable para la realización de simulaciones.

En el contexto global del desarrollo, la parte que ha correspondido al tema de este trabajo que se presenta en mayor detalle es:

- Configuración.
- Interfaz de usuario.
- Desarrollo ágil de nuevos usuarios, sistemas de recomendación, generación de usuarios...
- Extensibilidad.

1.4. Metodología

Este software se ha realizado en un grupo de varias personas, por lo que necesitamos de una metodología para organizarnos. Podríamos considerar que estamos utilizando Scrum[1], pero para los más puristas en cuanto a metodologías software no sería considerado como tal, ya que utilizamos una estructura organizativa horizontal, que suele ser más propio de empresas que venden su propio producto software o, como es el caso, en desarrollos de software para investigación. El equipo de desarrollo tiene un contacto directo con el cliente(que serían nuestros tutores de proyecto) y hay casi una comunicación total día a día con ellos, sin roles intermediarios. Sin embargo, sí que se tienen reuniones cada semana en el equipo para ver cómo avanza el proyecto, retrospectivas, prototipos, integración, pruebas... no obstante, como no aplicamos todas las reglas de Scrum, consideraremos que el desarrollo se está realizando con una metodología iterativa e incremental tal y como se muestra en la Figura 2

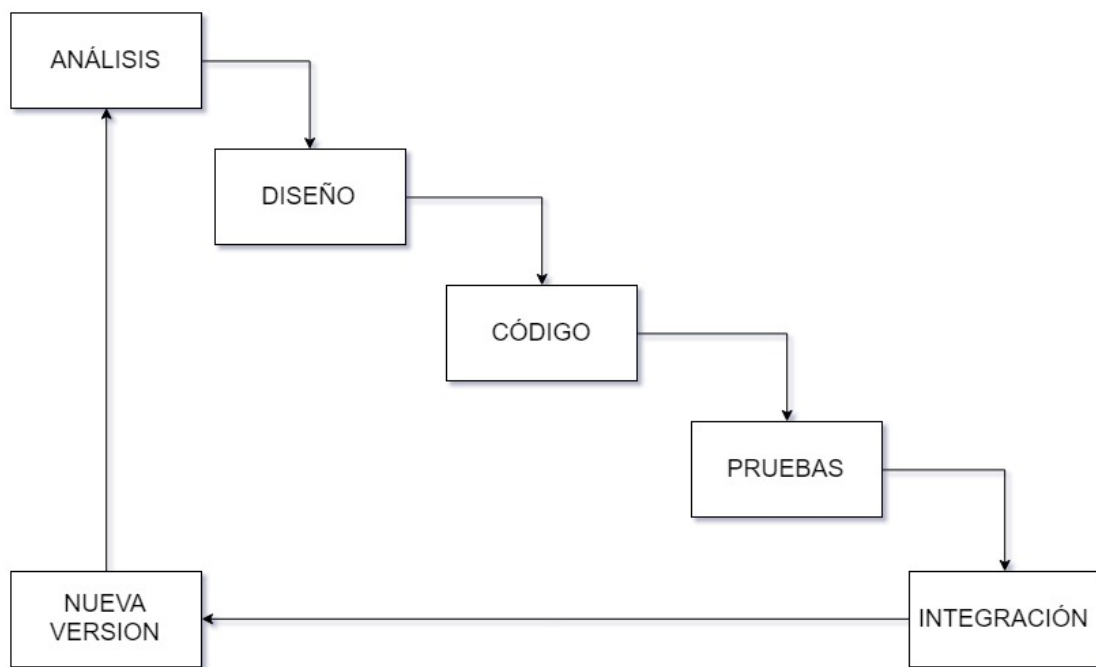


Figura 2: Ciclo iterativo e incremental

En este modelo, primero se realiza un análisis de los requisitos que se van a necesitar para cada iteración. Después del desarrollo de estos, se hacen pruebas y para finalizar se integra con el resto del sistema.

Cada 2 semanas realizamos una iteración donde se realizan todos los pasos comentados

anteriormente, donde todo el equipo de desarrollo decide qué requisitos son más críticos e importantes, bugs y releases. En base a estas decisiones y utilizando herramientas online como Trello³, se gestiona que tareas debe realizar cada uno.

Un desarrollo iterativo e incremental ofrece varias ventajas con respecto a otras metodologías como puede ser el desarrollo en cascada. Una de las ventajas que ofrece es la entrega de software usable a mitad de desarrollo, mientras que en el modelo en cascada cada fase del proceso debe ser finalizada (firmada) para pasar a la siguiente fase. El desarrollo de software no es lineal y esto crea dificultades si se utiliza una metodología en cascada[2].

Cada cierto tiempo realizamos una release. Utilizamos un formato de versiones semántico[3] del tipo X.Y.Z donde, X, Y y Z son números enteros mayores que 0.

³Es un tablero online donde se pueden crear, asignar y clasificar tareas, de tal modo que todo el equipo tiene una visión global del estado actual de desarrollo que se está creando.

2. Test

Prueba de referencia[4]

Referencias

- [1] J. S. Ken Schwaber, “La guía de scrum.” <https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf>, 2013.
- [2] I. Sommerville, *Ingeniería del software*. 2005, pp. 62–68.
- [3] “Semantic versioning 2.0.0.” <https://semver.org>.
- [4] J. Doe, *First book*. Cambridge: Cambridge University Press, 2005.