Sistema de interpretación de ensayos de bombeo en acuíferos

Informe

Cliente
Grupo de Hidrología Subterránea de Regional Norte

Tutores
Carla Forni
Pablo Gamazo

Estudiantes
Sebastián Daloia
Mathias Chubrega
Andrés Pías
Álvaro Correa
Jesús Guibert

Índice:

1 Introducción

1.1 Marco contextual

El presente proyecto es realizado para la obtención del título de Tecnólogo en informática de la Facultad de Ingeniería en convenio con la Universidad del Trabajo del Uruguay.

Su principal objetivo es la investigación y evaluación de diferentes tecnologías existentes y el desarrollo e implementación de un software para la interpretación de ensayos de bombeo de un acuífero. Los usuarios de este software serán los alumnos de una clase de hidrología subterránea, es por esto que a manera de introducción definimos a continuación su consistencia, en su más general concepción.

La hidrología subterránea es una ciencia que se encarga de estudiar las aguas subterráneas: su captación, circulación, sus condicionamientos geológicos y todos los fenómenos relacionados, además es muy importante para evaluar sistemas medio ambientales complejos. Actualmente los estudios hidrogeológicos son de especial interés para la provisión de agua a la población y para lograr una mejor preservación del medio ambiente. Aportan una mayor compresión del ciclo de vida de ciertos productos químicos y sustancias contaminantes, como por ejemplo, su movilidad, dispersión y la manera en que afectan al medio ambiente.

Un ensayo de bombeo consiste en colocar una bomba [1] en determinada parte del terreno, bombear un cierto caudal durante un tiempo determinado y medir los descensos en el nivel del agua que se producen en el pozo de bombeo y en otros pozos aledaños. Los ensayos de bombeo sobre una superficie permiten conocer determinadas condiciones sobre los suelos y el terreno.

Bajo este contexto, es que nuestro grupo del Tecnólogo informático, en forma conjunta con el grupo de Hidrología Subterránea, desarrollará un sistema enfocado en el diseño de una interfaz grafica amigable que brinde facilidades para el ingreso de datos y para la muestra de resultados a través de visualizaciones, y que permita procesar datos de campo, simular y predecir niveles de descenso del agua en todo el dominio para verificar determinadas condiciones de acuíferos genéricos.

1 Véase la sección Ensayos de bombeo, del capítulo “Conceptos generales sobre acuíferos” del documento “Estado del Arte”

1.2 Motivación

Es entusiasta la idea de integrar un grupo interdisciplinario que nos garantice la posibilidad de desarrollar nuestras habilidades de cooperación y comunicación con profesionales de otras áreas. Por otra parte, estamos frente a un hecho sin precedentes como lo es la colaboración en un proyecto entre la Regional Norte y el Centro Universitario de Paysandú. Esto puede significar un punto de partida hacia un mayor desarrollo de actividades integradoras futuras a ser realizadas conjuntamente entre ambos polos.

Es también para nosotros interesante el planteo que se nos hizo por parte del Ing. Pablo Gamazzo sobre la necesidad de desarrollar un producto inexistente como es la elaboración de un software para la interpretación de ensayos de bombeo que aporte información valiosa a sus usuarios.

De los ensayos de bombeo en acuíferos pueden obtenerse distintos parámetros que caracterizan al medio o al terreno en el que se encuentran. El conocimiento de dichos parámetros es indispensable tanto para optimizar la explotación del recurso a nivel local como para comprender el funcionamiento regional. Estos datos pueden resultar útiles para tomar decisiones en caso de que se quiera realizar algún tipo de explotación del terreno, como por ejemplo la tala, y no generar resultados nefastos para el medio ambiente.

1.3 Introducción al problema

El problema principal, en el estudio de los acuíferos, consiste en dar solución a la ecuación que gobierna el flujo de agua en un acuífero. La misma, es una ecuación en derivadas parciales y bajo ciertas hipótesis puede escribirse como:

$$S\frac{\partial h}{\partial t} = \nabla \cdot (\mathbf{T} \cdot \nabla h) + f$$

donde, T es la Transmisividad, S Coeficiente de almacenamiento, la variable h mide la variación del nivel de agua a lo largo del tiempo t y f es una constante.

El objetivo del trabajo no es generar soluciones para dicha ecuación, sino implementar un conjunto de algoritmos ya conocidos que le den solución, encontrando sus incógnitas a partir de datos ingresados. En cuanto a lo que respecta a la solución que desarrollaremos, la ecuación del problema resulta meramente anecdótica.

El proyecto consiste en desarrollar un programa para la interpretación de ensayos de bombeo. El sistema desarrollado deberá ofrecer a los usuarios una interfaz amigable que permita ingresar datos de un dominio que representará una porción de un acuífero subterráneo junto con todos los elementos involucrados.

El sistema deberá permitir al usuario ingresar las dimensiones del dominio, definir la ubicación de pozos de bombeo y de observación y las barreras en el plano y proporcionar datos de mediciones de campo para los pozos. Existen diversas soluciones analíticas y numéricas que permiten predecir los descensos del nivel de agua en función del caudal extraído y de los parámetros del acuífero, entre las cuales, el usuario podrá seleccionar una. Además será capas de seleccionar las subrutinas de optimización para calibrar los ensayos de bombeo.

Basándonos en nuestros intercambios con el Ing, Pablo Gamazo, sabemos que este software estará destinado básicamente a alumnos de Hidrología subterránea quienes lo utilizarían para experimentar con diferentes datos y así podrían ver el comportamiento que tiene un acuífero ante determinadas condiciones mediante gráficos en 2D y 3D. Cada alumno hará uso de esta aplicación desde su propia computadora. Los resultados obtenidos podrán ser almacenados en formato de video en la misma máquina del usuario.

Todos los algoritmos que involucren aspectos hidrológicos o de optimización serán aportados en código Matlab, por el Grupo de Hidrología Subterránea.

1.4 Objetivos

Desarrollar un producto de software que facilite el estudio e investigación del comportamiento de un acuífero luego de ser sometido a ensayos de bombeo presentando una interfaz muy amigable.

Ejercitar nuestra capacidad como investigadores en nuevas tecnologías: descubrir y evaluar diferentes herramientas tecnológicas, ya sean lenguajes o librerías que faciliten la tarea de desarrollo, sometiéndolas a evaluaciónes que nos lleven a determinar cuál es la mejor solución a implementar.

Es también nuestro objetivo, aprender y desarrollar capacidades de trabajo en equipo, realizando cada integrante su aporte personal al consenso general, concluyendo así en la culminación del trabajo.

1.5 Métodos de solución

El primer paso en el desarrollo de este proyecto consiste en el relevamiento de los requerimientos principales que deberá cumplir nuestra solución. Para esto, es necesaria una primera reunión con el cliente. De esta se desprende la necesidad de que la solución desarrollada cuente con una interfaz gráfica agradable dotada de gráficas de diferentes tipos, que posea un espacio de trabajo donde se puedan dibujar diferentes elementos geométricos, generar animaciones y exportarlas a video.

Debido a esto y a nuestra inclinación como grupo hacia la búsqueda de soluciones enfocadas desde la óptica informática, como segundo paso de la solución es necesario destinar esfuerzos a investigar dentro de diferentes lenguajes de programación y herramientas tecnológicas que permitan contemplar estos requerimientos.

Esto debe ser muy bien documento, ya que diferentes entornos son evaluados y comparados, por esto último, el tercer paso consiste en definir cuales herramientas informáticas serán realmente útiles para lograr concluir nuestros cometidos.

Debido al desconocimiento del contexto y el alcance del proyecto, como cuarto paso es necesario interiorizarse e investigar los diferentes temas que éste involucra. Por lo cual, se procede a recabar información referente a: aguas subterráneas en general, los acuíferos y sus diferentes tipos, niveles y superficies piezométricas, ensayos de bombeo y sus análisis, métodos de solución y algoritmos de optimización, entre otros; que permita una mejor familiarización con la temática del contexto mencionado.

Es así que dicha información es recabada en el documento del “Estado del Arte”, en el cual se reflejan los conceptos manejados y herramientas estudiadas a lo largo del proyecto. Luego de recopilar la información, el quinto paso es clasificar y definir que subconjunto de la misma nos resultará útil.

Los datos utilizados provienen de varias fuentes: sitios oficiales, blogs, foros, libros, etc. Sin embargo, cabe destacar que nuestra principal fuente fue la comunicación mantenida a lo largo del proyecto con el Grupo de la Regional Norte. A través del Ing. Pablo Gamazo, obtenemos gran parte de su solución actual desarrollada en Matlab [1] y se nos transmiten conceptos referentes a los diferentes mecanismos utilizados en el análisis de ensayos de bombeo. Además se evacuan nuestras dudas a través de varias entrevistas y se obtienen datos reales a ser utilizados para generar los diferentes casos de prueba de la aplicación.

El sexto paso consiste en el análisis de la solución. Desde un principio, sabemos que el sistema no estará definitivamente terminado al concluir este proyecto. Es por eso que debemos construir una solución que tenga la capacidad de ser extensible y crecer paulatinamente a través de la incorporación de mejoras y de nuevos ítems o algoritmos. Por lo cual, una vez que adquirimos una comprensión inicial del contexto y de nuestro problema a resolver, se vuelcan nuestros conocimientos en el desarrollo de una solución lo suficientemente genérica que garantice su posterior crecimiento y que abarque todos los requerimientos estipulados.

El séptimo y ultimo paso es el diseño e implementación de una aplicación multiplataforma que a través del entorno y herramientas seleccionadas, permita generar los mismos resultados ofrecidos por su sistema actual, pero brindando una interfaz mucho mas amigable con el usuario y que incorpore como eje central la posibilidad de representar y dibujar elementos geométricos a través de herramientas de dibujo.

Asimismo, se lleva a cabo una investigación sobre la herramienta científica Matlab para lograr una correcta comprensión de sus scripts para la traducción, en algunos casos, hacia el lenguaje utilizado en nuestra aplicación en desarrollo. Esto será útil además en el momento de la verificación del sistema.

Considerando que el proyecto debe involucrar un intercambio heterogéneo con el cliente, situado en la ciudad de Salto, se opta por una metodología de trabajo consistente en el desarrollo iterativo incremental del mismo, junto con la prototipación, de manera que ha medida que se avance en las diferentes etapas, se le entregue un prototipo descriptor del estado actual, el cual pudiese ser evaluado y correjido por esta parte interesada.

1 Véase el capítulo “Herramientas científicas existentes” de nuestro documento “Estado del arte” para mayor información sobre Matlab

1.6 Resultados

Los resultados obtenidos de este proyecto, muestran que fue posible alcanzar nuestros objetivos planteados inicialmente.

Como grupo hemos afrontado este desafío de manera conjunta, obteniendo buenos resultados en el desarrollo de un software de tipo científico. El hecho de abandonar el desarrollo de sitios webs y sistemas de facturación al cual estábamos habituados, nos mantuvo motivados a lo largo del proyecto.

Vivimos como grupo una experiencia productiva y novedosa, el tener que trabajar con un proyecto y cliente reales y una temática desconocida.

En proyectos realizados anteriormente, nos habían encontrado con letras de problema mucho más detalladas y sutilmente orientadas hacia un punto de vista informático. Asimismo, nuestros propios docentes nos fueron guiando hacia la resolución de nuestros propios problemas. Sin embargo, esta fue la primera vez que nos enfrentamos solos ante toda la problemática y ante el desarrollo de un software desde cero. En otras palabras, contamos con una letra básica que no fue escrita por una analista o ingeniero informático, en el que no se explicitaron detalladamente las funcionalidades que debía tener la aplicación.

Entendemos que en el desarrollo de este trabajo, nos fue bien, a pesar de que nos topamos con una serie de dificultades que se expresan mejor en la sección “Dificultades del proceso” del capitulo “5. Organización del proceso”. Principalmente afrontamos dos problemas que fueron: la falta de capacitación para desarrollar en la plataforma elegida y lo que mas nos costo fue lograr una correcta comprensión de la realidad. Sin embargo, fue posible sobrellevar esto de buena forma y durante el transcurso, todos aportamos lo mejor de nuestros esfuerzos.

Fue muy favorable tener una buena dinámica grupal, que originó un fortalecimiento como grupo y un crecimiento en cuanto a nuestras capacidades de comunicación y de trabajo en equipo. También mejoramos nuestra capacidad para interactuar de manera satisfactoria con profesionales de otras disciplinas y lograr beneficios para todas las partes involucradas.

Con la culminación de este trabajo, queda comprobado que es posible desarrollar a futuro nuevos proyectos conjuntos entre tecnólogos en informática y equipos de la Regional Norte. Por otra parte, que es posible la migración de una aplicación desarrollada en una herramienta científica dependiente de una plataforma hacia un ambiente de herramientas tecnológicas libres que ofrezcan una solución multiplataforma.

2 Definición del problema

2.1 Definición

Una vez conocido el contexto y las generalidades del proyecto, es pertinente especificar sobre los temas y conceptos correspondientes a la definición del problema.

En el estudio de los acuíferos y de sus parámetros hidrológicos intervienen conceptos matemáticos y físicos, a través del análisis del ingreso de parámetros del terreno es posible predecir los niveles de descenso en todo el dominio especificado, lo que se conoce como problema directo, por otra parte, también es necesario dar solución al problema inverso, que significa, partir de niveles observados para determinados tiempos y encontrar las propiedades del terreno que mejor se ajusten.
Los denominados métodos de solución abordan el problema directo, mientras que bajo el título de optimizaciones se encuentran los métodos que se encargan del problema inverso. Sabemos que en nuestro problema consideraremos un número reducido de métodos de solución y de algoritmos de optimización.

Estos problemas son bien conocidos por el grupo de Hidrología subterránea de la Regional Norte, quienes han implementado a través de scripts [1] de Matlab la serie de algoritmos para los problemas mencionados. Sin embargo, la solución existente no les permite la visualización gráfica, ni la manipulación del dominio y sus elementos. Los datos son proporcionados a través de la selección de opciones, similar a un shell script de Linux. Por otra parte, esta solución corre sobre Matlab, el cual es un software privativo, que está diseñado para generar scripts de corte científico.

Con base en esto, se plantea desarrollar un software científico que, aplicando conocimientos hidrológicos e informáticos, pueda brindar una solución acorde a los requerimientos del cliente y que facilite el estudio de estos fenómenos. Esto se consigue mediante el intercambio de información con nuestros pares de la Regional Norte, lo que conducirá indefectiblemente a una importante investigación previa al desarrollo.

Este marco se centra sobre varias lineas de trabajo: poder conocer en detalle y de forma mas acertada la realidad y temática del problema, estructurar una solución coherente y lo suficientemente genérica que permita su posterior extensibilidad y añadir mas parámetros, mas soluciones o mas algoritmos al problema, como también brindarle al usuario una buena experiencia de uso de la interfaz gráfica del sistema.

A continuación en el punto 2.2 se observan los cambios de requerimientos a lo largo del desarrollo del proyecto. Mientras que en el punto 2,3 se observan los requerimientos finales, tanto funcionales y como no funcionales.

1 Véase la definición de script en el capítulo “Glosario” del documento “Glosario, apéndice y bibliografía”

2.2 Evolución de requerimientos

Desde el inicio el objetivo fundamental del proyecto fue el desarrollo de un producto de software para la facilitación del estudio e investigación del comportamiento de los acuíferos, con énfasis en los tipos de acuíferos existentes en nuestro país.

Dada la distancia física con el cliente, y el detalle mayúsculo de comprender los conceptos técnicos de la hidrología, se generó durante todo el proceso una comunicación bidireccional mediante envío de correos electrónicos, y la concreción de entrevistas virtuales.

Lo anteriormente descrito fue el mayor determinante en la evolución de los requerimientos.

Como principio nuestra tutora Carla Forni nos presentó un documento impreso bajo el título: “Desarrollo de un programa de interpretación de ensayos de bombeo en un acuífero”, el cual nos daba las primeras pautas en cuanto a objetivos y requerimientos. A continuación la lista de requerimientos inicial:

- Crear proyecto.
- Ingreso de dimensiones del dominio a representar.
- Selección de método solución, e ingreso de parámetros asociados al método.
- Creación del dominio.
- Ingreso o exportación de caudal bombeado.
- Ingreso o exportación de observaciones.
- Ingreso gráfico o mediante datos de puntos en el dominio.
- Asociación de datos a puntos en el dominio.
- Ingreso gráfico o mediante datos de rectas en el dominio.
- Asociación de datos a una recta en el dominio.
- Selección de algoritmo de optimización.
- Visualización de resultados.
- Generación de animaciones.
- Generación de gráficos 2D.
- Creación de base de datos para manipulación de datos de campo y simulación.

En suma, se debía crear mediante interfaz gráfica, un nuevo proyecto con determinadas dimensiones para un dominio, sobre el cual se pudiese ubicar los diferentes elementos involucrados. A continuación, se podría importar datos sobre caudales u observaciones, los cuales serían agregados a una lista, con un nombre como identificador. También se podría comenzar a ubicar elementos en el dominio. Luego de cumplido estos pasos, se comenzaría con la asociación de datos a los elementos del dominio. Se seleccionarían un método de solución y un algoritmo de optimización para posteriormente ver gráficas y animaciones resultantes del análisis de los datos. En principio los datos de campo y de simulación generados debían de ser almacenados en un medio persistente.

El día Jueves 22 de Noviembre del 2011 se llevó a cabo la primera reunión entre las partes interesadas, en nuestra ciudad Paysandú, disertante Ing. Pablo Gamazo.

Esta fue decisiva como manera de esclarecer el problema y los conceptos del dominio de esa realidad. Se pulieron varias ideas generales, y del posterior análisis se concluyó por actualizar los requerimientos de la siguiente manera. La creación del proyecto, el ingreso de las dimensiones del dominio y la selección del método solución pasaron a considerarse como uno solo. A su vez, se indicó que el ingreso de elementos al dominio no debía de reparar en una escala correspondiente con alguna realidad, con lo cual el usuario sería el arbitrario en la colocación de los mencionados elementos en el dominio.

Gamazo en su exposición clarificó el comportamiento de cada gráfica y las diferencias entre la de optimización con las demás; ahora se debían de considerar algoritmos de optimización y visualización de sus resultados por un lado, y métodos de solución y sus resultados por otro, también dada la simplicidad de representación se optó por incluir una gráfica 3D para estos últimos.

A su vez del debate surgido se entendió como prescindible la utilización de una base de datos, ya que el carácter de cada proyecto sería temporal y los datos de campo y simulación podían ser sin problemas manejados en memoria, se concluyó por persistir únicamente en un formato de video las gráficas generadas.

A partir de esta reunión la identificación de requerimientos fue tomando una forma más específica, reconociéndose los siguientes, y sus funcionalidades.

1. Creación de un nuevo proyecto, consistiendo en ingresar el alto y ancho del dominio sobre el cual se va a trabajar y un método de solución. Además se deben ingresan los parámetros principales que tendrá el dominio como por ejemplo, los valores para S y T.

2. Definir dominio, se genera un plano en dos dimensiones donde se irán ingresando y modificando los pozos y barreras. Esto podría ser hecho mediante coordenadas o con la interacción gráfica. Los elementos geométricos que se deben utilizar son rectas y puntos.

3. Definir los pozos de bombeo. Cada pozo tendría asociada una colección de quíntuplas de datos de la forma (x,y,Q,ti,tf) donde x,y indica la posición, Q es el caudal y ti,tf son los tiempos inicial y final.

4. Definir los pozos de observación.

5. Definir las barreras del dominio.

6. Ingresar o importar caudales de bombeo, consiste en el ingreso manual, o mediante ficheros con cierta estructura, de datos asociados al caudal bombeado en determinado pozo.

7. Ingresar o importar observaciones de ensayo, consiste en el ingreso manual, o mediante ficheros con cierta estructura, de datos asociados las observaciones hechas en determinado pozo.

8. Asociar datos a pozos de observación, consiste en la tarea de asociar datos de observación a dichos pozos.

9. Asociar datos a pozos de bombeo, consiste en la tarea de asociarles datos de bombeo a dichos pozos.

10. Visualización de resultados y animación. Luego de la aplicación del método solución a través de los datos y los parámetros ingresados al momento de la creación del proyecto se representarán diferentes gráficas en dos y tres dimensiones. Las mismas deberán poder ser pausadas, rebobinadas, exportadas, y aumentada su velocidad de reproducción. El sistema debe permitir simular la variación del nivel del agua (nivel piezométrico) [1] para escenarios genéricos utilizando las diversas soluciones existentes. Los resultados a visualizar serán superficies equipotenciales en el plano similar a las curvas de nivel [2] que representen los niveles de descenso (h) producidos en todo el dominio.

11. Visualizar gráfica de optimización, se generará una gráfica a partir de las función objetivo de optimización en función de parámetros del problema y de las observaciones.

El 26 del Setiembre, se le solicitó al cliente que nos indicara cuales serían los datos de entrada que manejaría la aplicación.

En esta ocasión nos envía un archivo que nos fue útil para determinar el formato y el tipo de datos que el usuario ingresaría al sistema para proporcionar los caudales y las observaciones a utilizar. Llegamos a la conclusión de que el sistema debía manipular internamente el resultado de la aplicación de un método de solución en una matriz de tres dimensiones: dos para determinar la ubicación en el plano.

Para las observaciones y los bombeos, los valores podrán ser ingresados por parte del usuario, o ser suministrados a través de un fichero de extensión txt o de una hoja de cálculo de Open Office [3] donde los datos vienen divididos en columnas.

En un nueva entrevista con nuestro cliente el día 4 de Noviembre, en una etapa temprana del análisis, recibimos la siguiente información.

En el analisis de ensayos de bombeo [4], existen diversos métodos de solución analíticos o numéricos que permiten predecir la variación del nivel piezométrico en función del caudal extraído y los parámetros del acuífero.

1 Ver la sección 1.3 Nivel piezométrico del capítulo “Conceptos generales sobre acuíferos” del documento “Estado del Arte”
2 Ver la sección 1.3.1 Superficie piezométrica del capítulo “Conceptos generales sobre acuíferos” del documento “Estado del Arte”
3 Véase la definición de Open Office en el capítulo "Glosario" o para mayor información consulte la referencia 44 del capítulo “Referencias” en el documento “Glosario, apéndice y bibliografía”
4 Véase la sección Análisis de ensayos de bombeo del capítulo “Conceptos generales sobre acuíferos” del documento “Estado del Arte”.

Se consideró como primer metodo de solucion al Theis [1] , del cual conocemos que tiene cinco parámetros: Theis(r,t,Q,T,S). Los parámetros que puede recibir son:

r: distancia radial desde el pozo de observación y el pozo de bombeo.
t: tiempo medido en días.
Q: es el caudal del pozo el cual se mide en m^3/d, por ejemplo 500 m^3/d.
T: transmisividad [2] son m^2/d, por ejemplo 1000 m^2/d.
S: coeficiente de almacenamiento, por ejemplo 0.0001, determina el descenso en todo el dominio.

El Lunes 14 de Noviembre, en nuestra primera entrevista por Skype [3], se introdujeron varios nuevos conceptos.

El método de solución es el problema directo, en el cual se tienen los valores de los parámetros, por ejemplo T y S y a partir de estos se calcula el descenso en todos los puntos del dominio (se obtienen los h).

Debíamos considerar dos tipos de soluciones en nuestro sistema: analíticas y numéricas. Dependiendo de cual seleccionara el usuario, se introducirían determinas condiciones a cumplir en cuanto a parámetros y elementos del dominio para ofrecer resultados fehacientes. Las características de las diferentes soluciones analíticas y numéricas son las siguientes:

Analíticas

- No todas permiten o soportan el uso de barreras [4].
- Se les puede pedir el valor de la variable para cualquier tiempo, o sea que la variable se obtiene a través de una función (cuya complejidad es relativa).

Numéricas

- Hay que definir una discretización espacial, es decir el usuario decide sobre cómo dividir el dominio.
- Hay que definir una discretización temporal: definir tiempo inicial y final y longitud de paso de tiempo.
- La variable se obtiene de una matriz que se calcula para cada paso de tiempo.

El algoritmo de optimización permite resolver el problema inverso al que es resuelto por el método solución. Se parte de algunos valores de h observados y se prueba con diferentes T y S para encontrar lo que dé más parecido.

Otro concepto introducido fue que en el sistema no diferenciáramos los pozos de observación de los de bombeo. A un mismo pozo se le puede asociar el caudal bombeado, así como también observaciones hechas en el mismo.

1 Ver la sección Método Theis, del capítulo “Conceptos generales sobre acuíferos” del documento “Estado del Arte”.
2 Ver el capítulo “Conceptos generales sobre acuíferos” del documento Estado del Arte
3 Aplicación orientada a la comunicación interpersonal y multimedia a través de Internet.
4 Véase en este capítulo la descripción de barreras.

Identificamos que Definir dominio era un requerimiento central del proyecto, ya que por este medio se harán las asociaciones a los pozos, se determinará la existencias de barreras, y su distribución por el dominio. Las gráficas trabajarán en torno a los datos que allí estén manifiestos.

La siguiente reunión se produce en el mes de Diciembre de 2011 cuando efectuamos la primera entrega de la documentación

Gracias a esta reunión mejora nuestra compresión de la realidad y se elabora una nueva versión del modelo de dominio. Hasta ahora habíamos identificado los diferentes parámetros hidrológicos (T y S) como propiedades del terreno que estaban asociadas al dominio. Al explicarsenos que cada método de solución considerara distintos parámetros, se modifica la clase “Solución”, asociandole su propia lista de parámetros.

También se nos explicó mas sobre las barreras y las condiciones externas, además sobre los diferentes tipos de gráficas que el usuario debería visualizar, esta información originó cambios en los casos de uso respectivos.

Se nos enviaron los primeros scripts de Matlab, donde estaban implementados el método Theis, el algoritmo Calitheis y un módulo de prueba. A partir de esto, obtuvimos la siguiente información.

CaliTheis2 es un algoritmo de optimización específico de la solución analítica de Theis, del cual conocemos que tiene 8 parámetros:
CaliTheis2 (Q,obs,r_obs,t_obs,Tmin,Tmax,Smin,Smax)

Los parámetros que puede recibir CaliTheis2 son:

Q: caudal

obs: conjunto de datos correspondiente a las observaciones obtenidas, luego de haber aplicado a cada uno de sus elementos(observaciones) la solución analítica Theis.

r_obs: conjunto de datos relacionados con obs, indicando la distancia radial desde un pozo de observación y uno de bombeo para cada elemento de obs.

t_obs: conjunto de datos relacionados con obs, indicando el tiempo en días para cada elemento de obs.

Tmin: Transitividad Mínima

Tmax: Transitividad Máxima

Smin: Coeficiente Mínimo

Smax: Coeficiente Máximo

Los valores Tmin,Tmax,Smin,Smax son aquellos que el usuario ingresa para probar combinaciones entre ellos para encontrar los valores de T y S más óptimos. Esto datos nos ayudaron a mejorar la compresión de la funcionalidad de aplicación de Algoritmos de optimización.

Se nos explicó como tenia que aplicarse el cálculo de Theis, que en un principio se pensó efectuarlo de forma puntual. Hasta ese momento, no supimos que teníamos que considerar niveles iniciales.

Pensábamos que esto se tenia que efectuar solamente para los tiempos y niveles asociados con el pozo de bombeo. Luego los resultados para todo el dominio se pensaban extender por interpolación, sin embargo, esto no es correcto, para graficar la evolución del dominio, los cálculos se deben aplicar a una grilla de puntos cuyas dimensiones y naturaleza no fueron especificadas completamente en ese momento.

Por otra parte, nuestro tutor había omitido contarnos de que existía una ecuación que regía los niveles iniciales que va a tener el agua en todo el dominio: Ho [1]. Esto obligó a modificar el caso de uso de Nuevo proyecto, agregando varios ítems para el ingreso de tres constantes que determinarían los coeficientes de la ecuación del plano Ho.

El siguiente hito importante en nuestro proyecto se produce el 6 de enero con la entrega de nuestro primer prototipo.

En una nueva reunión, nuestro cliente nos introduce el requerimiento de poder hacer Zoom sobre el dominio, o sea, la posibilidad de acercarse y alejarse en visualización, es por esto, que tras las limitaciones de la clase QPainter, utilizada para dibujar los elementos del dominio en el espacio de trabajo, fue que se optó por buscar un framework orientado a objetos: Qgraphics, lo que permitió incorporar además de lo mencionado, una mejor dinámica de manipulación y posicionamiento de elementos geométricos.

A través del envío de un script de Matlab, que contenía básicamente todas las funcionalidades de la aplicación, se agregaron varias consideraciones para el cálculo de métodos de solución. La presencia de más de un pozo de bombeo en el dominio y de una barrera introducía cambios en este requerimiento para todas las soluciones. Se reformularon varios aspectos de nuestra aplicación.

Selección de Discretizaciones: Se introdujo la selección por parte del usuario de discretizaciones espaciales y temporales y la utilización de las mismas para los diferentes cálculos y visualizaciones posteriormente generadas. En este aspecto tuvimos una evolución de nuestro entendimiento del problema, tras la implementación de este requerimiento logramos visualizar resultados coherentes a partir del procesamiento de la información; a su vez, abandonamos nuestra idea original de efectuar un cálculo puntual sobre cada uno de los tiempos y caudales asociadas al pozo de bombeo correspondiente.

1 Ver el apartado “4.8.2 Implementación de cálculos” del capitulo “4. Solución Informática” del presente documento.

Se corrigió la aplicación de métodos de solución y en particular el cálculo de Theis; se agregaron, a este requerimiento, los aspectos referentes a la generación de datos para el gráfico del campo de velocidades lo cual fue adquirido del script enviado (uso de gradientes) y se transcribió a nuestra aplicación ya que inicialmente no se había hecho un exhaustivo diseño de este caso de uso por falta de información referente a este tipo de gráficos.

El 13 de enero se produce el envío de una nueva versión de la documentación y junto a ello una nueva reunión

Se modifica la funcionalidad de importar e ingresar datos de bombeo para considerar la utilización de un “postproceso” [1] y de “pozos virtuales” [1]. No sabíamos ni el como ni el porque los datos de los caudales bombeados debían ser procesados internamente para trabajar con otros valores posteriormente.

Se nos aclaró que esto se debía a algo llamado "principio de superposición de efectos" [1] que se da cuando los problemas son lineales, este principio permite sumar los efectos de los diferentes fenómenos; los métodos de solución implementados sirven para un único pozo bombeando un caudal constante, por lo tanto, cuando un pozo cambia de caudal lo que el sistema debe hacer internamente es agregar otro pozo que produce el efecto contrario.

Se modificó el cálculo interno de los métodos de solución ya que al añadir una barrera al dominio, es necesaria la creación de tantos pozos virtuales como pozos de bombeo existan, los cuales se ubicarán en determinadas coordenadas dependiendo de los coeficientes de esta recta. Este proceso se debe cumplir con el fin de satisfacer el principio de superposición de efectos mencionado.

El 2 Febrero enviamos un nuevo prototipo de la aplicación mucho mas avanzado y junto a ello se produce una reunión por Skype.

Al ver nuestro prototipo, nuestro tutor nos envió una nueva versión de su script de Matlab, donde agregaba un nuevo método de solución con un nuevo algoritmo de optimización. Además nos indicó que la aplicación de estos últimos podía involucrar más de un pozo de observación.

Nos advierte que para la visualización de las gráficas de descensos en todo el dominio debíamos abandonar nuestra visión inicial donde estas se presentaban todas juntas., nos explica que un usuario de la aplicación preferiría que las diferentes visualizaciones se le muestren por separado, tal cual puede verse en el script enviado.

Los cambios descritos implicaron la modificación de las correspondientes funcionalidades de nuestra aplicación.

1 Ver el apartado “4.8.2.2 Postprocesos” del capitulo “4. Solución Informática” del presente documento

2.3 Requerimientos

2.3.1 Requerimientos no funcionales

El sistema debe estar preparado para operar en múltiples sistemas operativos, como mínimo en Windows y distribuciones GNU/Linux.

Debe ser una aplicación de escritorio, ya que todo servicio web ha sido destacado como prescindible.

Ningún dato ha sido remarcado esencialmente como persistente, por lo tanto no se debe elegir medios de persistencia. Lo único que puede ser almacenado en el sistema del usuario es un video con las animaciones generadas.

Otro requerimiento es que el sistema desarrollado sea extensible. Una vez que el sistema esté funcionando, deberá ser posible agregar nuevos métodos de solución o algoritmos de optimización.

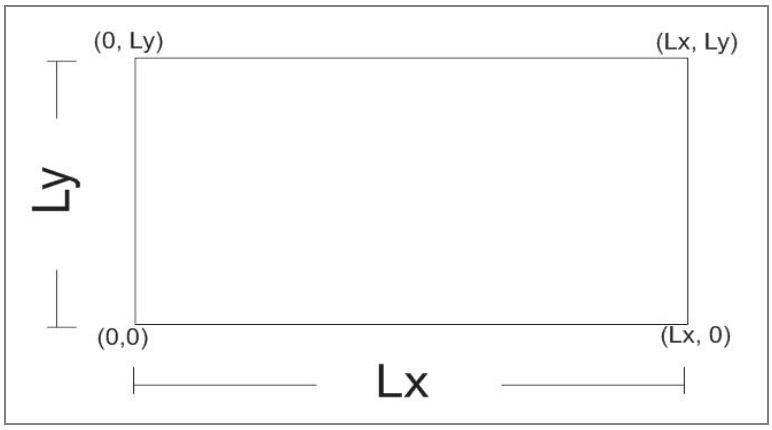
2.3.2 Requerimientos Funcionales

El sistema debe cumplir los siguientes requerimientos que permitirán que el usuario logre una correcta interpretación de los ensayos de bombeo.

2.3.2.1 Creación de un nuevo proyecto

Este requerimiento permitirá ingresar un conjunto de datos de entrada y resultados para un problema dado.

Deberá permitir al usuario definir las dimensiones del dominio a representar ingresando valores numéricos para Lx (ancho) y Ly (Alto). A partir de esta información se generará gráficamente el siguiente recuadro a través del cual será posible definir los elementos del dominio como pozos y barreras.



Además, a través de este requerimiento será posible definir la ecuación del plano que regirá los niveles iniciales (Ho) en todos los puntos del plano a través del ingreso de tres coeficientes: a,b,c. Toda esta información luego será utilizada para los cálculos de los métodos de solución.

2.3.2.2 Métodos de solución

En nuestro sistema existirá, en un principio, un número determinado de métodos de solución. Más adelante, se van a ir generando nuevos métodos que se implementaran en el mismo.

Cada método encapsulará su propia lógica de cálculos. Esto significa que si bien para todos los métodos los cálculos se realizaran sobre todo el dominio, la generación puntual de resultados será responsabilidad de cada uno de ellos. El usuario podrá seleccionar un método de solución del tipo analítico o numérico, y además deberá ingresar todos los parámetros o condiciones del acuífero asociados al mismo, como por ejemplo los valores para la Transitividad y el Coeficiente de Almacenamiento. Cada método deberá tener asociado sus propios parámetros.

2.3.2.3 Ingresar o importar caudales de bombeo

Este requerimiento implica que el usuario podrá suministrar los datos del caudal bombeado por una bomba de manera manual o a través de un fichero, asignándole un nombre. El usuario tendrá la posibilidad de ingresar de forma “manual” uno a uno los tiempos y los caudales de bombeo y así registrar un ensayo de bombeo en el sistema. Opcionalmente, podrá importarlos desde un archivo. El fichero podrá ser de texto (txt) o de Open Office Calc (ods) y deberá cumplir un formato particular.

Esta información será almacenada en memoria, en estructuras con caudal en función del tiempo, para luego ser asociada a su correspondiente pozo en un paso posterior.

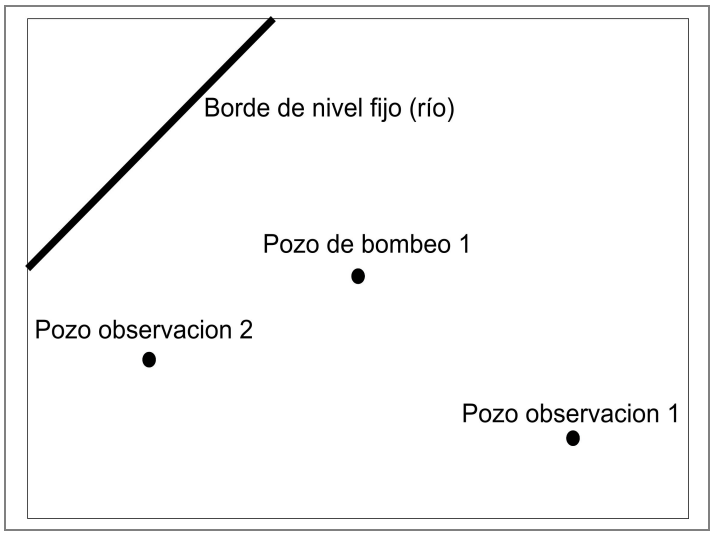
2.3.2.4 Ingresar o importar observaciones del ensayo

Este requerimiento garantiza que el usuario podrá suministrar un conjunto de observaciones del ensayo, de manera manual o a través de un fichero, asignándole un nombre. El usuario tendrá la posibilidad de ingresar de forma “manual” una a una las observaciones, sus tiempos y niveles observados y así registrar un nuevo conjunto de observaciones en el sistema. Opcionalmente, estos datos podrán ser importados desde un fichero del tipo texto (txt) o de Open Office Calc (ods) y que deberá cumplir un formato particular.

Esta información será almacenada en memoria, en estructuras con nivel en función del tiempo, para luego ser asociada a su correspondiente pozo en un paso posterior.

2.3.2.5 Definir dominio

Este es el requerimiento central del proyecto y consiste en ofrecer una interfaz 2D para definir el dominio. Se permitirá el ingreso de planos, rectas y puntos para representar los diferentes elementos como los pozos y las barreras; los pozos de bombeo o de observación se representaran gráficamente mediante puntos y se ingresarán a través de una herramienta gráfica o a través de sus coordenadas, las barreras, ya sean bordes impermeables[1] o de nivel fijo[2], se representarán mediante rectas y se ingresarán mediante una herramienta gráfica o a través de sus datos (coordenadas de dos puntos).



Una vez que cada pozo es ubicado, se le podrá asociar las observaciones o bombeos cargados previamente.

2.3.2.5.1 Barreras

En el dominio pueden existir barreras positivas o negativas. Las barreras positivas son por ejemplo un río o un arroyo, las negativas son por ejemplo montañas; estas van a influir de diferente manera para dar solución al proyecto.

Las barreras afectarán a todos los métodos de solución. En términos matemáticos, deben ser consideradas como cualquier otro de los parámetros o propiedades del dominio (S, T, etc.).

Existirán además de estas barreras, condiciones externas o de contorno, las cuales se definirán sólo para las soluciones numéricas no para las analíticas (como por ejemplo si hay un lago fuera del dominio porque en esos lugares el nivel no va a cambiar). Para el dominio se podrá definir dos tipos de condiciones de contorno: de nivel o de flujo;[3] a efectos prácticos esto quiere decir que el usuario le asignará a cada contorno un tipo y luego un valor.

1 Borde impermeable: Como su definición lo da a entender, es una barrera impermeable. En los ensayos de bombeo se da el fenómeno de que el nivel de descenso es mayor del lado de estos bordes, ya que llega menos agua.
2 Borde de nivel fijo: Ríos o lagos, cuyo nivel no se ve afectado por el ensayo de bombeo.
3 Las condiciones de contorno delimitan el comportamiento del flujo del agua en el transcurso aguas abajo.

2.3.2.6 Asociar datos a pozo

Esta funcionalidad permitirá al usuario finalmente enlazar un punto del dominio con un conjunto de observaciones o de bombeos cargados previamente en el sistema. Como paso previo a esta etapa, el usuario debe haber definido al menos un punto en el dominio y debe haber cargado al sistema un conjunto de observaciones o de bombeos.

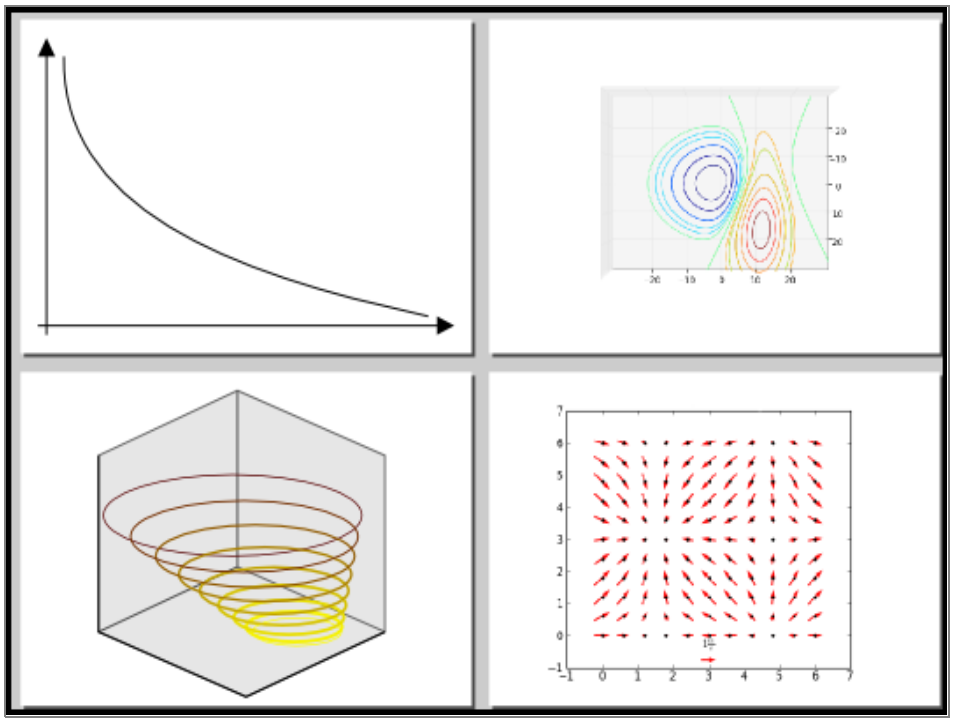
Un mismo pozo en el dominio podrá ser de bombeo o de observación, por lo cual no es correcto diferenciarlos según su tipo.

2.3.2.7 Visualización de resultados y Animación

Esta funcionalidad permitirá al usuario visualizar simulaciones de la variación del nivel de agua a partir de los resultados obtenidos tras aplicar el método de solución sobre los datos y parámetros de entrada. Es así que se generarán gráficas animadas para representar los niveles de descenso (h) producidos en todo el dominio, en dos o tres dimensiones (a través de curvas de nivel) o gráficos vectoriales en dos dimensiones para representar el campo de velocidades. También se graficarán las observaciones.

Las animaciones deben de manera dinámica:

- Poder ser pausadas
- Poder ser reproducidas en reversa
- Seleccionar la velocidad de incremento de tiempo
- Exportar el video.



2.3.2.7.1 Gráficas

Dependiendo de que esté calculando el usuario se graficarán diferentes cosas. Por esto, existirán los siguientes casos de gráficas:

Caso 1: Gráficas animadas de los ensayos de bombeo, que se generan a partir de la aplicación de un método de solución.

En este caso se simula el ensayo de bombeo, considerando valores fijos para los parámetros hidrologicos (T, S, etc). Por lo tanto, el usuario deberá visualizar solamente los valores de los descensos en todo el dominio (la gráfica de los niveles de agua en el plano 3D y 2D) o el campo de velocidades (gráfico vectorial 2D).

Caso 2: *Gráfica 2D con el valor de los descensos vs. tiempo en puntos de observación.*

2.3.2.7.2 Visualización

Se necesitaría una representación visual para todo el dominio y una por pozo. También una de la función objetivo si se hace optimización. Se deberá generar una animación para la representación del dominio.

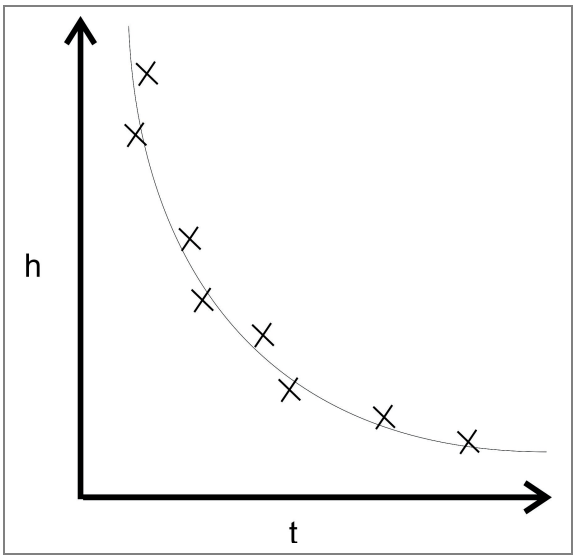
2.3.2.8 Algoritmos de Optimización.

La optimización puede involucrar uno o más pozos de observación. El algoritmo más sencillo es correr el mismo caso para varias combinaciones de parámetros. Existen otros más complejos que se basan en minimizar la diferencia entre lo observado y lo medido. Cada algoritmo deberá tener asociado los propios parámetros que utiliza.

Estos algoritmos serán encargados de encontrar los parámetros hidrológicos que mejor ajusten una función objetivo con las observaciones de los ensayos. El usuario podrá seleccionar y ejecutar un algoritmo de optimización para cada pozo de observación existente.

2.3.2.9 Visualizar gráfica de optimización.

Para cada pozo de observación se generaran gráficas en dos dimensiones de la función objetivo de optimización en base a los parámetros del problema y a las observaciones. En la siguiente imagen se ve cómo se grafican conjuntamente la función objetivo y las observaciones.



En esta etapa, el sistema resuelve el problema inverso. El usuario elige un método de optimización que busca encontrar la mejor solución aproximada y lo que se deberá graficar es la diferencia entre los niveles observados y los niveles medidos en la etapa anterior para diferentes tiempos.

Se grafican, en 2D, juntas, la función objetivo[1] y los puntos que representan las observaciones en el transcurso del tiempo.

1 Ver definición de función objetivo en Estado del Arte, punto 1.8 Algoritmos de Optimización.

3 Solución al problema

3.1 Introducción

Comenzando con la solución a los requerimientos del problema, en primera instancia se investigó sobre los diferentes lenguajes y tecnologías existentes, evaluando y teniendo en cuenta que deberían sernos útiles para el desarrollo del proyecto.

Tuvimos que encontrar herramientas que nos permitiesen brindarle al usuario una interfaz gráfica en dos dimensiones, que fueran útiles para dibujar elementos geométricos en una “pizarra” o espacio en blanco y que luego de un determinado procesamiento de datos se visualizaran gráficas animadas en dos y tres dimensiones de los resultados. Estas herramientas además debían permitir exportar la animación generada en formato de video.

A partir de esto, fue posible llevar a cabo una comparativa y selección de tecnologías.

Luego se prosiguió con la investigación sobre los diferentes conceptos enfocados a los acuíferos subterráneos y a los ensayos de bombeo ya que teníamos presente la importancia de familiarizarnos con los distintos temas a abordarse a lo largo del proyecto para, de esta manera, lograr una mejor comunicación utilizando una jerga común con el grupo de Hidrología Subterránea, quienes fueron nuestros guías en la temática del problema y ademas nuestros clientes para este proyecto.

A lo largo de todo el proyecto mantuvimos un intercambio permanente con este grupo lo que nos ayudó a lograr una correcta comprensión de la realidad del problema, la cual fue madurando y creciendo.

En una siguiente etapa procedimos a dividir la complejidad del problema original en otros más pequeños determinados por las funcionalidades del sistema. Identificamos casos de uso críticos y no críticos, cada integrante de nuestro equipo se hizo responsable de trabajar en sus propios casos a lo largo de todo el proyecto. A este punto nos encontramos en el análisis, donde iríamos generando nuestras particulares inquietudes entorno a las funcionalidades a ofrecer en el sistema.

Es por esto que se fueron elaborando una serie de entrevistas con Pabo Gamazo por mail o Skype a medida que avanzábamos en cada etapa, buscando evacuar las dudas surgentes. Como se fundamenta en el apartado “Arquitectura” del capítulo “Solución informática”, nuestro sistema no utiliza persistencia, por lo cual no se diseñó ningún modelo de entidad relación. Los diferentes conceptos del problema fueron manipulados como clases.

Nuestra visión inicial e individual del problema luego fue volcada en el modelo de dominio, el cual es una representación visual estática del entorno real, objeto del proyecto. Este fue creciendo o cambiando a medida que aumentaba también nuestro conocimiento del tema u obteníamos mayor información sobre el problema. Por ejemplo, en determinado momento se nos explicó que existía una ecuación del plano

que regia los niveles iniciales de agua en todos los puntos del dominio. El modelo de dominio fue elaborado en conjunto.

Para el modelado del dominio se tuvo en cuenta crear una solución lo suficientemente genérica y extensible; garantizando que el sistema pueda dar solución a los diferentes problemas tratados sobre acuíferos de características muy genéricas y que sea posible su extensión a medida que se agreguen nuevos métodos, algoritmos, o cambien determinadas condiciones de los problemas tratados. Teniendo en cuenta esto y según lo discutido con el grupo y Pablo Gamazo, se consideró definir una clase "Solución", de la que desciendan dos clases mas: “Analítica” y “Numérica”, para luego ir haciendo especializaciones de estas. De la misma forma que a los métodos de solución, se consideraron a los algoritmos de optimización; en otras palabras se diseñó una clase “Optimización” donde todos los algoritmos hereden de ésta.

Continuando con la solución, pasamos a los diagramas de secuencia del sistema y el diseño de las vistas, más adelante, comenzamos a codificar.

En esta etapa, nos encontramos capacitándonos en la tecnológica Phyton. A su vez, cada uno tuvo que investigar en detalle aquellas bibliotecas que utilizaría y concentrarse en resolver sus propios problemas: performance, usabilidad, amigabilidad. Por citar un ejemplo, quien tuvo que dedicarse a la parte de graficado, tuvo que aprender a utilizar la librería correspondiente para lograr generar gráficas de diferentes tipos y de forma rápida. Por eso recurrimos, con base en la previa investigación, a consultar sitios webs oficiales, blogs y foros sobre estas tecnologías.

Como es sabido este sistema constará con un número reducido de casos de uso muy interrelacionadas entre sí, ya que para la ejecución de una funcionalidad se exige que previamente el usuario haya ejecutado otra serie de pasos anteriores, por lo cual fue necesario trabajar de a pares para acoplar casos de uso contiguos y de esta manera lograr la integración de la aplicación. Se verificó que cada funcionalidad entregara a las demás los resultados esperados.

Durante este proceso, rendimos cuenta a nuestros compañeros mediante documentación y prototipación temprana. A través de las devoluciones recibidas fue posible aclarar conceptos, determinar los requerimientos necesarios y desarrollar un sistema acorde a las necesidades de nuestro cliente.

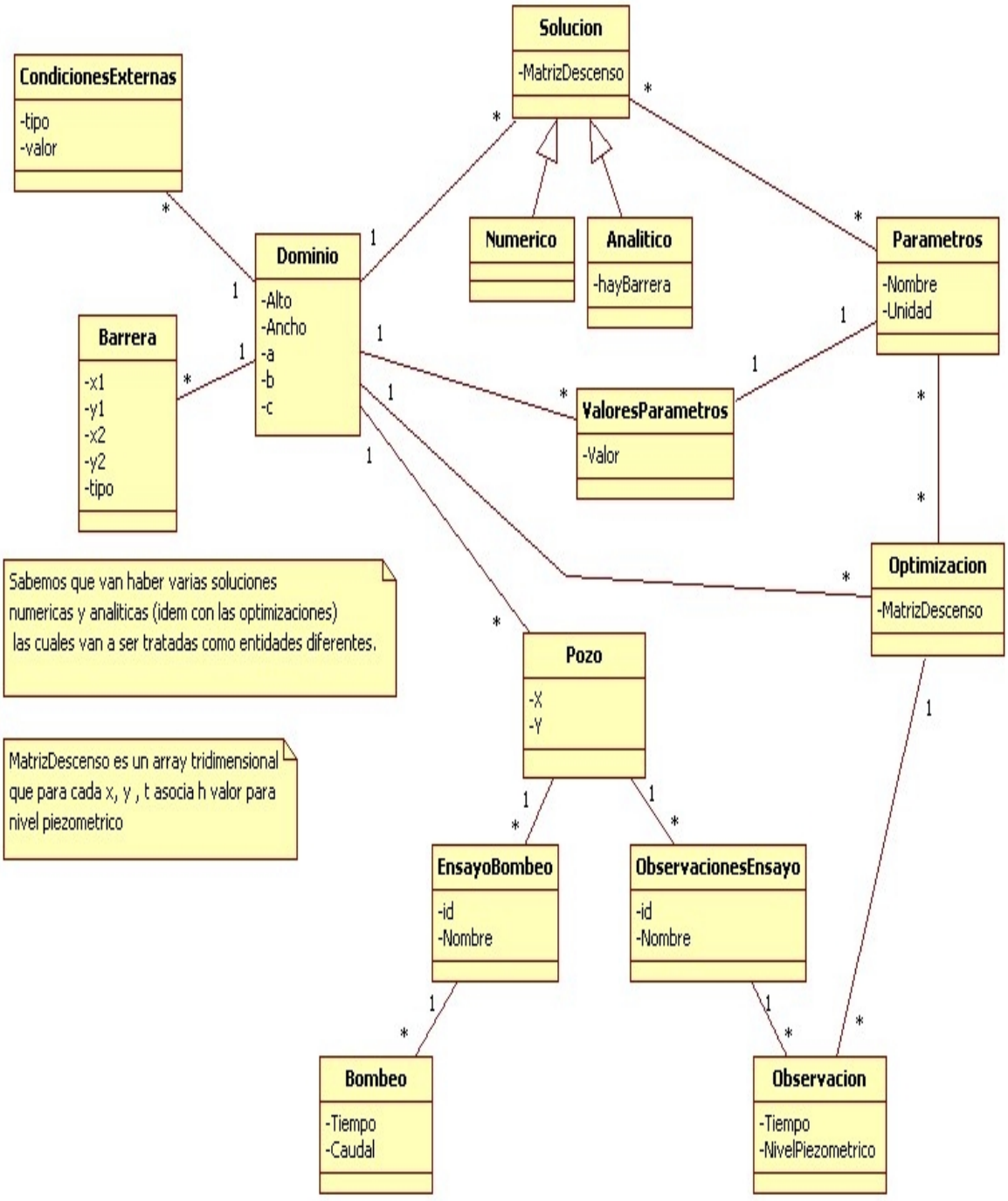
Como se explica en el capítulo “Organización del Proceso”, la constante en este proyecto fueron las iteraciones, nuestra visión inicial fue atacar el problema básico de un acuífero con características muy genéricas, partimos de implementar un único método de solución considerando un único pozo de bombeo en el sistema. Con cada incremento, se fueron agregando más parámetros y conceptos a nuestro sistema para dar lugar paulatinamente a una solución más compleja.

Gracias a la ayuda de los scripts que nos fue enviando Pablo Gamazo, nuestra solución fue creciendo y se fue agregando mas lógica al problema. Se incorporaron aspectos como mas pozos de bombeo, mas pozos de observación, consideraciones para la interacción con barreras. A medida que avanzamos con la investigación e íbamos madurando la solución descubrimos que la forma en que manipularíamos y

procesaríamos los datos cambiaba. Esto además nos fue útil para verificar que nuestro sistema se ajustara realmente a la solución existente desarrollada por el equipo de Hidrología subterránea, la que les ofrecía resultados exactos, lo cual permitió un refinamiento de nuestra solución.

La tareas de documentación, así como la de comunicación y comprensión de la realidad se mantuvieron a lo largo de todo el proyecto.

3.2 Modelo dominio



En el análisis del modelo de dominio se tuvieron en cuenta los siguientes aspectos. Existirán varios parámetros a ser utilizados por los diferentes métodos de solución del sistema, los cuales podrán ser compartidos por más de una solución, sin embargo, los parámetros y sus valores no deben ser almacenados o instanciados en el sistema dos veces ya que las propiedades del terreno serán las mismas para un mismo proyecto. En otras palabras, se sebe impedir que para cada método se realice nuevamente una asociación de valores para sus parámetros si esta ya fue realizada, para así evitar la redundancia de información en el sistema de manera innecesaria.

Por estos motivos y por el hecho de que en un futuro pudieran coexistir dos instancias de métodos de solución diferentes dentro de un mismo proyecto se consideraron a los Parámetros (hidrológicos) como una entidad separada en el sistema.

Se consideró separar los nombres de los parámetros de sus valores para facilitar, en el momento de crear un nuevo proyecto, el despliegue al usuario de un listado de las casillas que debe completar de acuerdo al método seleccionado. Para lo cual se hace necesario instanciar parámetros que no poseerán valores inicialmente; esto con lo mencionado al principio de este punto, va a permitir que los valores para cada parámetro en el sistema se instancien y asocien una vez y estén disponibles o puedan ser compartidos por más de un método de solución en el sistema.

Por esto mismo, la clase dominio estará vinculada con los valores asociados a esos parámetros, de tal forma que estos datos estén disponibles en el caso de la instanciación de un nuevo método. De esta forma, cada método de solución operará con sus propios parámetros y de él se podrá conocer la cantidad que manipula. Del dominio, se sabrá además la cantidad de valores de parámetros hidrológicos que posee asociados.

Esta estructura garantizara a futuro la extensibilidad del sistema, ya que inicialmente se tendrán harcodeados[1] todos los parámetros que dispondrá el sistema para usar en los diferentes métodos de solución, lo cuales serán accedidos de forma genérica a través de identificadores numéricos o cardinales. Esto garantizara que en una etapa posterior, en que sea necesario agregar un nuevo método de solución al sistema que utilice nuevos parámetros no considerados previamente, sea más simple de implementar, cargando un nuevo parámetro a la lista inicial e indicando al constructor del método de solución los cardinales de los parámetros a utilizar.

1 Término de la jerga informática, ver Glosario, Hard-Code

4 Solución informática

4.1 Introducción

A continuación se presenta la solución informática empleada para este proyecto, listando los casos de uso y resumiendo los críticos. También se analiza el entorno a operar, junto con la arquitectura más adecuada para la solución, la argumentación sobre las herramientas elegidas y se presentan diagramas de secuencia del sistema y de casos de uso. Se cubren características sobre el diseño, implementación y testing del proyecto.

4.2 Casos de uso

4.2.1 Listados de requerimientos

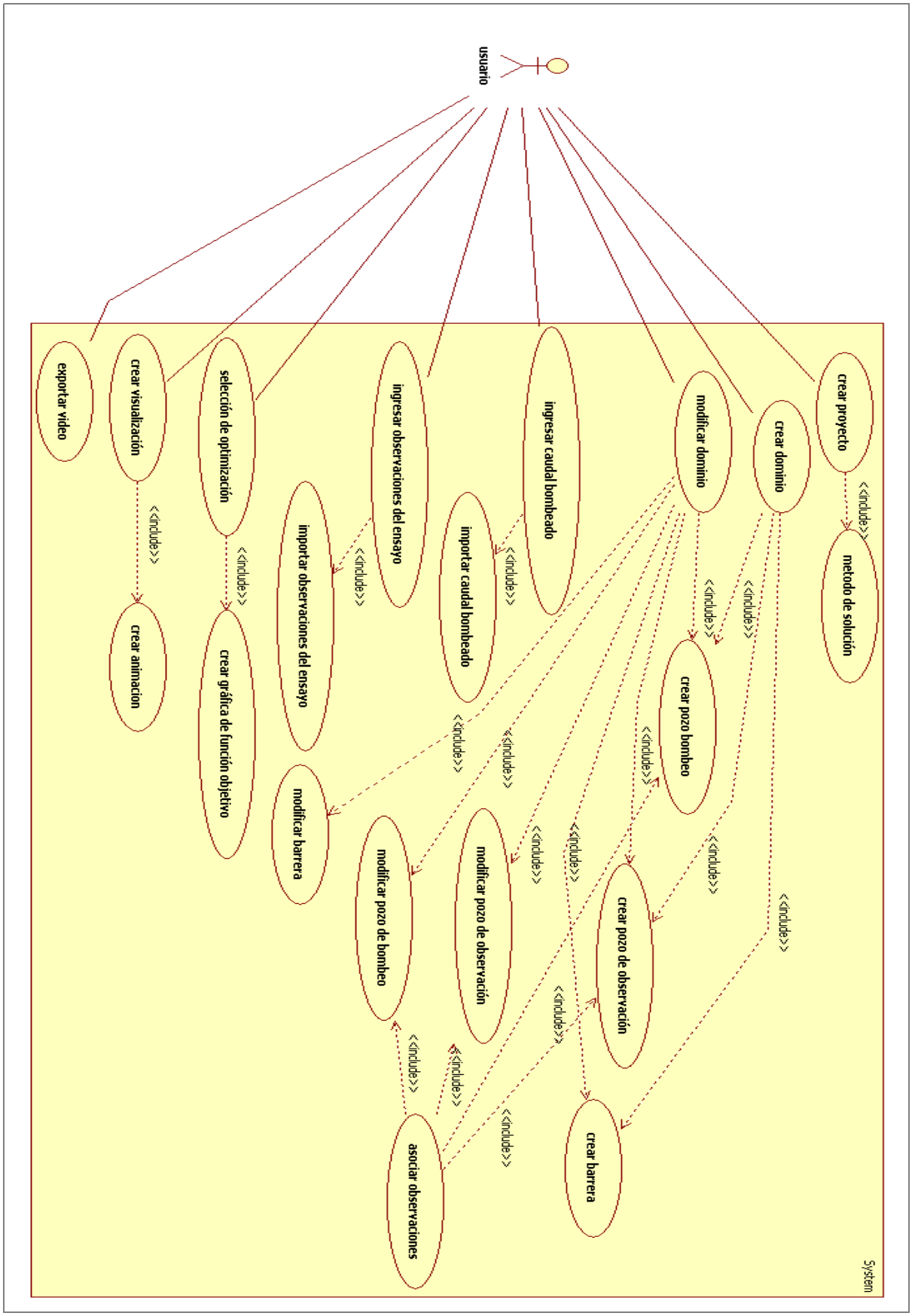
4.2.1.1 Casos de uso críticos

- Crear proyecto
- Definir dominio
- Ingresar caudales para pozos
- Ingresar observaciones para pozos
- Crear visualización
- Seleccionar algoritmo de optimización

4.2.1.2 Casos de uso no críticos

- Modificar dominio
- Importar caudal bombeado
- Importar observaciones del ensayo
- Exportar video

4.2.2 Diagrama de casos de uso



4.2.3 Descripción de los casos de uso críticos

4.2.3.1 Crear Proyecto

Este caso de uso nos permite dejar sentadas las mínimas bases para comenzar a trabajar. Es el primer paso a seguir siempre que queramos trabajar en el Programa. Primero el usuario deberá seleccionar las dimensiones del dominio con el cual va a trabajar; también ingresará los Parámetros del Dominio: Coeficiente de almacenamiento y Transmisividad. Seleccionará además el Método de Solución a utilizar e ingresará los parámetros requeridos por dicho Método. En caso de ser un Método de Solución de tipo Numérico podrá además indicar las Condiciones Externas.

4.2.3.2 Definir Dominio

Se enfoca en la creación gráfica de los diferentes elementos que van a componer el dominio, posibilitando el dibujo y posicionamiento de pozos y barreras, asociándoles coordenadas y datos de los ensayos realizados.

4.2.3.3 Ingresar caudal bombeado

Este caso de uso permite ingresar los datos de los caudales bombeados por los pozos de bombeo a medida que transcurre el tiempo. El usuario deberá ingresar para cada bombeo que desee registrar: un valor en segundos para indicar el tiempo en que se produce y el caudal bombeado en ese momento. Una vez que el usuario intente dar de alta estos datos, se chequeará que el tiempo sea superior al tiempo del bombeo ingresado previamente. Una vez que complete el procedimiento, el usuario podrá confirmar que se almacene la información, con lo cual el sistema solicitará un nombre y a continuación registrará un nuevo ensayo de bombeo, o podrá desechar los datos ingresados.

4.2.3.4 Ingresar observaciones del ensayo

Este caso de uso permite ingresar los caudales observados por los pozos de observación a medida que transcurre el tiempo. El usuario deberá ingresar para cada observación que desee registrar, un valor en segundos para indicar el tiempo en que se produce y el nivel piezométrico alcanzado en ese momento. Una vez que el usuario intente dar de alta estos datos, se chequeara que el tiempo sea superior al tiempo del nivel piezométrico ingresado previamente. Una vez que complete el procedimiento, el usuario podrá confirmar que se almacene la información con lo cual el sistema solicitará un nombre y a continuación registrará un nuevo conjunto de observaciones, o podrá desechar los datos ingresados.

4.2.3.5 Crear visualización

Esta funcionalidad permitirá al usuario crear un conjunto de cuatro gráficas a partir de los datos previamente cargados en el sistema. Además, tres de estas gráficas serán animadas, por lo que se provee de controles para reproducir, pausar y exportar las gráficas en formato de video.

4.2.3.6 Selección de optimización

Este caso de uso permite realizar la asociación entre un pozo del dominio y un método de optimización. El usuario indicará dentro del dominio un determinado pozo y qué método de optimización utilizará. Estos datos de asociación (pozo-optimización) serán almacenados, pero se permite al usuario modificar las asociaciones creadas.

4.3 Entorno a Operar

El entorno [1] a operar es otro de los requerimientos en un proyecto ya que delimita qué tipo de arquitectura se tiene que implementar, y establece los tipos de subsistemas de la misma.

Se distinguen, por sus diferencias de carácter, dos tipos de entornos de desarrollo: el web y el de escritorio.

4.3.1 Entorno Web

Para el común de las aplicaciones, el entorno web tiene las siguientes ventajas. Es multiplataforma [2] logrando funcionar bajo cualquier sistema operativo. Es portable [3] ya que incrementa el nivel de los dispositivos consumidores de la aplicación. Consume pocos recursos, debido a que el procesamiento queda a cargo de un servidor y además ocupan poco espacio. Sus desventajas son que se debe destinar un ordenador como servidor y que no es posible acceder al Hardware local directamente para el procesamiento.

Dadas sus características se puede obtener una solución viable al problema, la cual implementaría un sistema con arquitectura web cliente/servidor [4] donde la aplicación se encontraría alojada en un servidor brindando los distintos servicios a los cuales accederían los clientes.

4.3.2 Entorno de escritorio

El sistema es creado para ejecutarse en una computadora de escritorio sobre un sistema operativo. A consecuencia de esto, el sistema desarrollado puede generar un lazo permanente sobre el sistema operativo (dependiendo para cual fue desarrollado) no pudiendo migrar la solución a otra plataforma [5]. Otra desventaja frente al entorno web, es que una modificación en la aplicación implica que la misma sea replicada nuevamente en todos las computadores donde esté insalada. Por el contrario el entorno de escritorio no depende de un computador oficiando de servidor, por lo cual la necesidad de conección a alguna red es prescindible; una aplicación ejecutándose en este ambiente puede acceder al hardware, permitiendo efectuar operaciones de entrada y salida con los perifericos del computador.

Considerando sus características, y por el motivo de que como requerimiento no funcional se destaca su desempeño sin la necesidad de conectarse a la red o internet, la viabilidad del entorno de escritorio para esta aplicación es certera.

1 Entorno: donde la aplicación se desarrolla, entre los más importantes se encuentran el entorno orientado a web y el de escritorio. El primero orientado a operar en un sistema distribuido en donde coexiste una comunicación a través de una red de comunicaciones; el segundo orientado a operar en un sistema monolítico, donde trabaje bajo una terminal sin necesidad de comunicarse con otros.
2 Multiplataforma: capacidad de un sistema de software en operar igualmente en distintas plataformas
3 Portabilidad: Facilidad de un programa de ser ejecutado en diferentes ambientes, con distintos sistemas operativos.
4 Véase la definición en el capítulo “Glosario” del documento “Glosario, apendice y bibliografía”
5 Consideración importante al momento de elegir el lenguaje de programación.

4.3.3 Entorno elegido

Acorde a la cuestión sobre qué entorno utilizar, los nombrados anteriormente cumplen los requisitos para generar una solución. La elección de un entorno de escritorio surge en el momento en que el cliente especifica que la aplicación pueda ejecutarse sin requerir un servidor. Este requisito es válido porque el cliente no considera necesario disponer de un servidor ni de una conexión directa con este. Así mismo no descarta que a futuro pueda ser una funcionalidad que se pueda integrar en el producto final.

4.4 Arquitectura del sistema

4.4.1 Arquitectura

Dentro del ciclo de vida [1] que lleva el desarrollo del software, una de las pautas más importantes a tener en cuenta es la correcta elección de la arquitectura del sistema [2].

4.4.2 Estilos arquitectónicos

Desde el principio se planteó que unos de los objetivos importantes que debía tener el estilo arquitectónico era la de dividir las responsabilidades de la aplicación en un sistema de entidades que posean identidad y cooperen entre ellas. Es cuando se integró la arquitectura orientada a objetos [3], aportando enormemente la elaboración del modelo de dominio (técnica DDD) [4]. Es de señalar que estos objetos se comunican por medio de llamadas a métodos o acceso a propiedades de otros objetos, utilizando interfaces bien definidas.

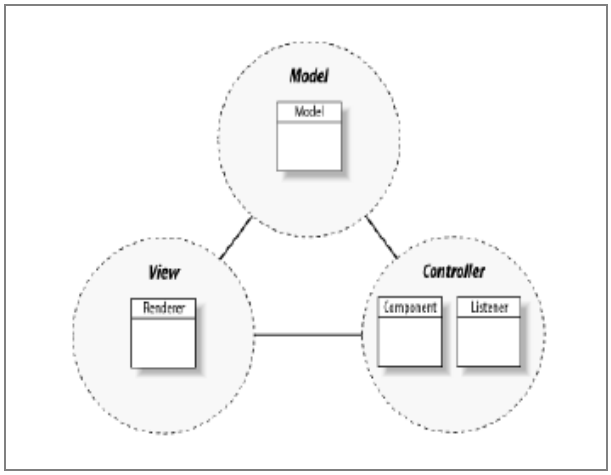
La complejidad de la arquitectura orientada a objetos nos llevó a la elección de integrar otra arquitectura que coopere con la anterior, y que nos incremente el nivel de abstracción de los objetos. Es por eso que incluimos la arquitectura en capas [2].

La arquitectura en capas agrupa las funcionalidades relacionadas en una aplicación: en capas que se apilan verticalmente (vista como una pirámide invertida), donde cada capa agrega las responsabilidades y abstracción de su predecesora ubicada inmediatamente debajo de ella.

1 Véase la referencia 35 del capítulo “Referencias” del documento del documento “Glosario, apendice y bibliografía”
2 Véase la referencia 36 del capítulo “Referencias” del documento del documento “Glosario, apendice y bibliografía”
3 Véase el apéndice 2.5 del capítulo “Apéndice” del documento del documento “Glosario, apendice y bibliografía”
4 DDD: El Diseño Guiado por el Dominio (DDD) es un enfoque orientado a objetos para diseñar software a partir de los objetos del dominio, con sus comportamientos y relaciones. El software construido es una proyección directa del modelo.

4.4.3 Patron MVC (Model View Controller)

Para seguir un orden de la arquitectura en capas, implementamos el patrón MVC, el cual nos divide la aplicación en tres capas bien definidas:



1 Modelo: contiene la lógica de negocio
2 Vista: muestra los datos al usuario de la aplicación y permite su interacción; un modelo puede tener asociada más de una vista, dependiendo de cómo se quieran mostrar los datos.
3 Controlador: es la capa que interactúa entre las capas Vista y Modelo; recibe los pedidos de los usuarios, desde la vista, realiza las operaciones necesarias en el modelo y retorna una respuesta a la vista.

De esta forma se simplifica el desarrollo y mantenimiento de la aplicación, ya que cada componente tiene claramente delimitado su campo de acción.

4.5 Elección del lenguaje y Herramientas

Dentro de la amplia gama de lenguajes existentes para este entorno, se tomó la decisión de evaluar mediante puntos específicos los siguientes: Java, C/C++ y Phyton, en donde se buscó encontrar fortalezas y debilidades de cada uno de estos, para determinar cuál se adecuaba más a una solución satisfactoria.

También se relevó información sobre herramientas de graficado, y de generación de videos.

De la comparativa surgió la inclinación por un lenguaje: Python, una herramienta para graficado: Matplotlib, y una interfaz gráfica: PyQt4. A continuación se resume la elección hecha sobre estas tres áreas de la investigación.

4.5.1 Python

Es un lenguaje con una extensa y buena documentación, fácil de aprender, cuya sintáxis ayuda a los programadores a escribir de forma simple y prolija.

La ventaja fundamental es la existencia de un framework para graficado de funciones en dos y tres dimensiones: Matplotlib el cual como se ha explicado [1] surge a partir de las limitaciones de desarrollo de la herramienta Matlab. Esta acotación se debe a que nuestro proyecto se basa en una versión codificada en esta última herramienta.

La base matemática del proyecto es abarcada por dos librerias, numpy y scipy, la primera es la librería principal para la computación científica con Python, y la segunda, sirve fundamentalmente para trabajar con arreglos n-dimensionales, proveyendo eficientes rutinas para manipulación numérica.

4.5.2 Matplotlib

Es un framework de Python para graficado y procesamiento de datos, los argumentos favorables ha esta herramienta son los siguientes.

Existe un conjunto amplio de funciones compatibles con Matlab, por lo cual resulta sencillo migrar la aplicación ya desarrollada actualmente hacia este lenguaje (matplotlib-mlab). Además si el día de mañana se desea generar un nuevo cambio en algún script Matlab, no habría mayores problemas para pasarlo a Python.

Posee la potencia de hacer muchas cosas con poco código, ya que con unas pocas líneas es posible generar muy buenos resultados, en dos o tres dimensiones. Es fácil de programar.

Es una librería con amplia documentación que permite, además de generar las gráficas en diferentes dimensiones, generar animaciones de forma nativa, sin necesidad de codificar programas externos o utilizar librerías también externas.

1 Véase investigación en “Estado del Arte” sobre herramientas de graficado, “MatPlotLib”.

Matplotlib tiene su propio framework para generar animaciones usando FFMPEG[1] o Mencoder[2], de forma nativa. Soluciona por si sola la parte de las animaciones. No es necesario utilizar otras librerías como processing. Ya incluye soporte para las gráficas, animaciones y videos.

La comparación entre matplotlib y otras herramientas, de otros lenguajes, investigadas arrojan las siguientes evaluaciones.

Jzr3d [3], librería de Java, no genera por si misma videos, siendo necesario llamar a la librería Processing [4] para poder generarlos. Además, en la etapa de investigación y prueba hubo que hacer varias configuraciones y copiar varias librerías para que funcionara en Windows, un proyecto de ejemplo en Netbeans [5].

En C/C++ primero se debió elegir la interfaz de usuario, Qt, la cual demandó varios pasos de configuración. Su librería nativa Cairo sólo genera gráficas en dos dimensiones, por lo cual como paso ulterior se debió investigar sobre alguna librería de graficado que permitiese expresar las exigencias del proyecto. Por el contrario en Python sólo se necesitó instalar las herramientas sin paso previo alguno de configuración.

4.5.3 Interfaz gráfica

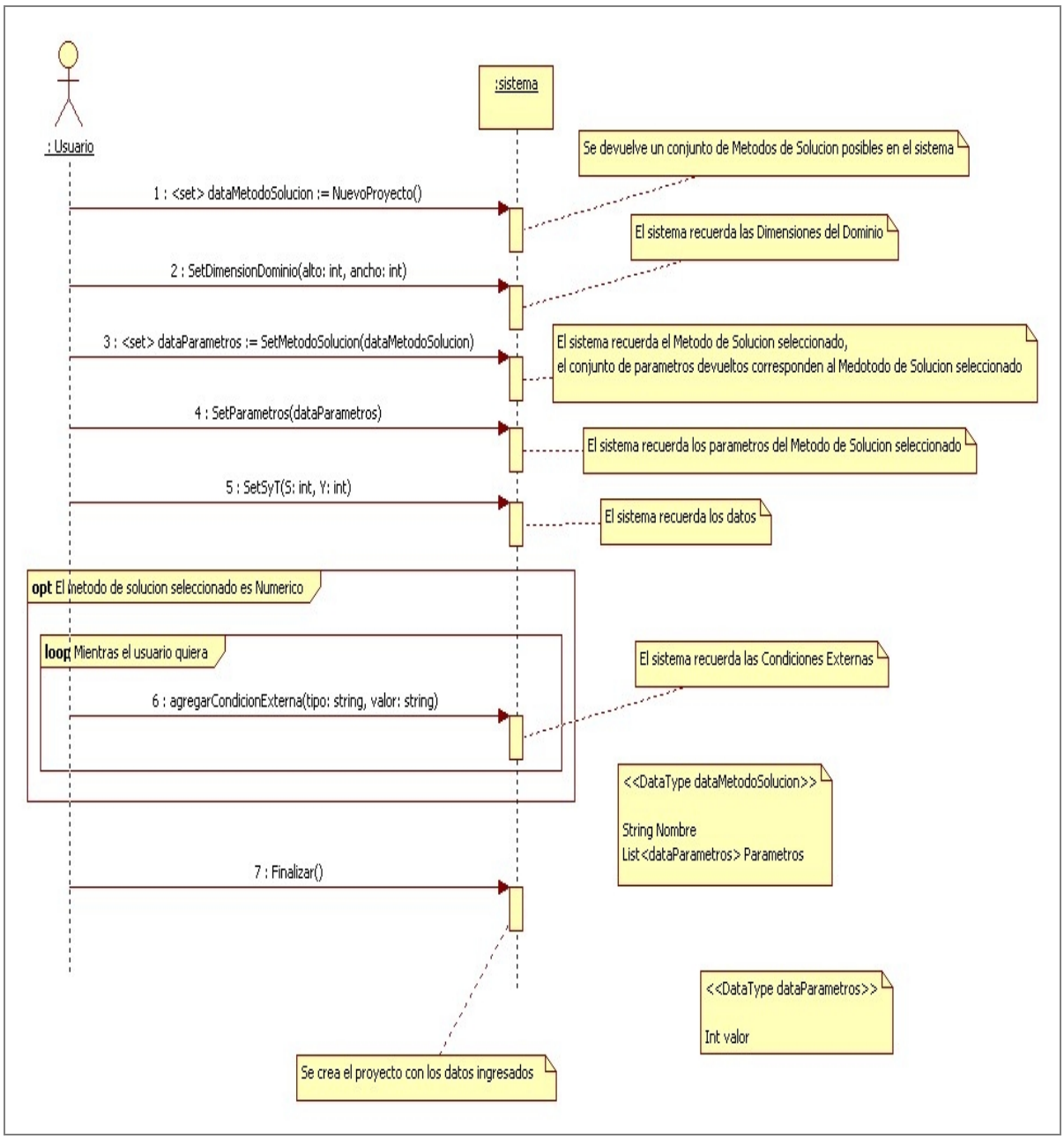
La interfaz elegida para el lenguaje que optamos es PyQT [6]. Se trata de una herramienta capaz de generar aplicaciones de escritorio a medida, rápidas y robustas. La ventaja de Python es la capacidad de utilizar archivos que estén escritos en C o C++, los cuales son utilizados por los sistemas operativos para escribir extensiones de sus módulos, o por librerías de interfaz gráfica que codificadas en este lenguaje corren como código nativo en cada máquina. Esto hace que la respuesta ante la creación y los eventos de la interfaz gráfica sea rápida. En C/C++ la configuración de Qt para utilizarlo requiere bastantes más pasos y en Java la complicación reside en que todos los elementos de sus interfaces gráficas están escritos en Java, es decir cada píxel es dibujado en pantalla mediante su máquina virtual, lo cual hace notablemente lento el desarrollo de la interfaz mediante SWING[7] u otro framework GUI de Java.

1 Véase en “Estado del Arte”, investigación sobre Codificadores de videos, FFMPEG.
2 Véase en “Estado del Arte”, investigación sobre Codificadores de videos, Mencoder.
3 Véase en “Estado del Arte”, investigación sobre Java, Jzr3d.
4 Véase en “Estado del Arte”, investigación sobre Java, Processing.
5 Framework para desarrollar Java, en Windows.
6 Véase descripción de QT y PyQT.
7 Framework para desarrollar interfaz gráfica en Java.

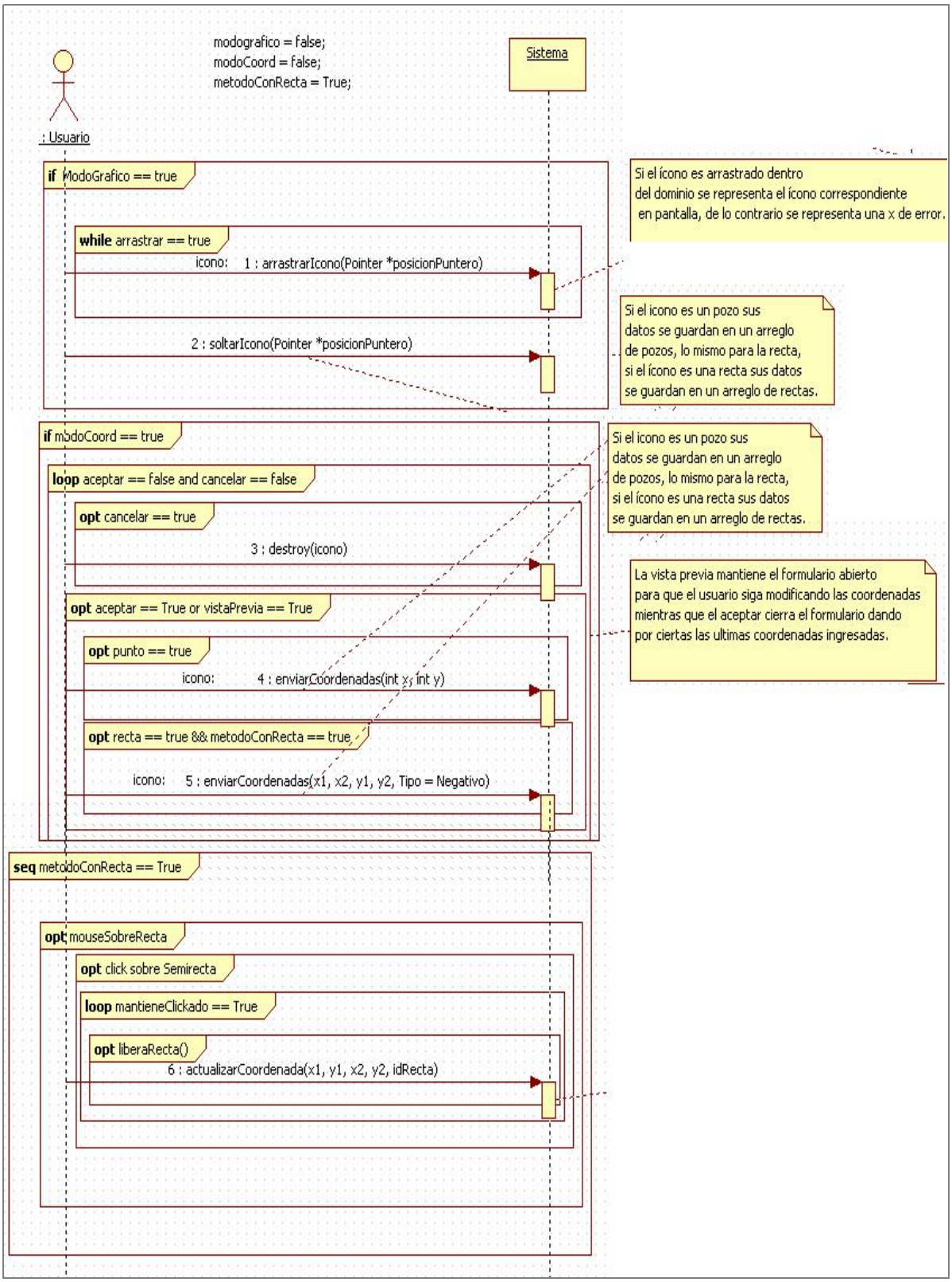
4.6 Diagramas de secuencia del sistema

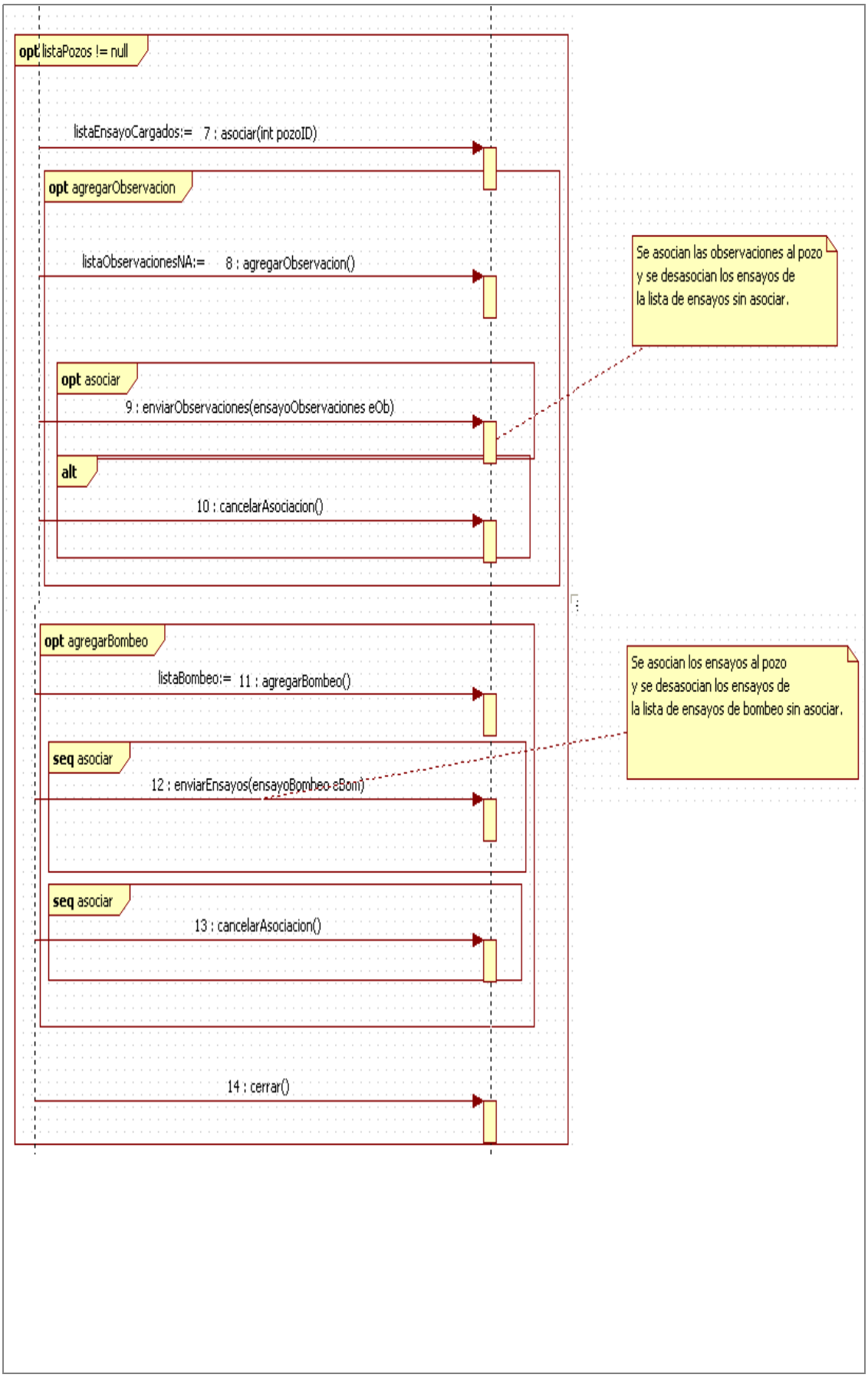
Se presentan los diagramas críticos para el sistema.

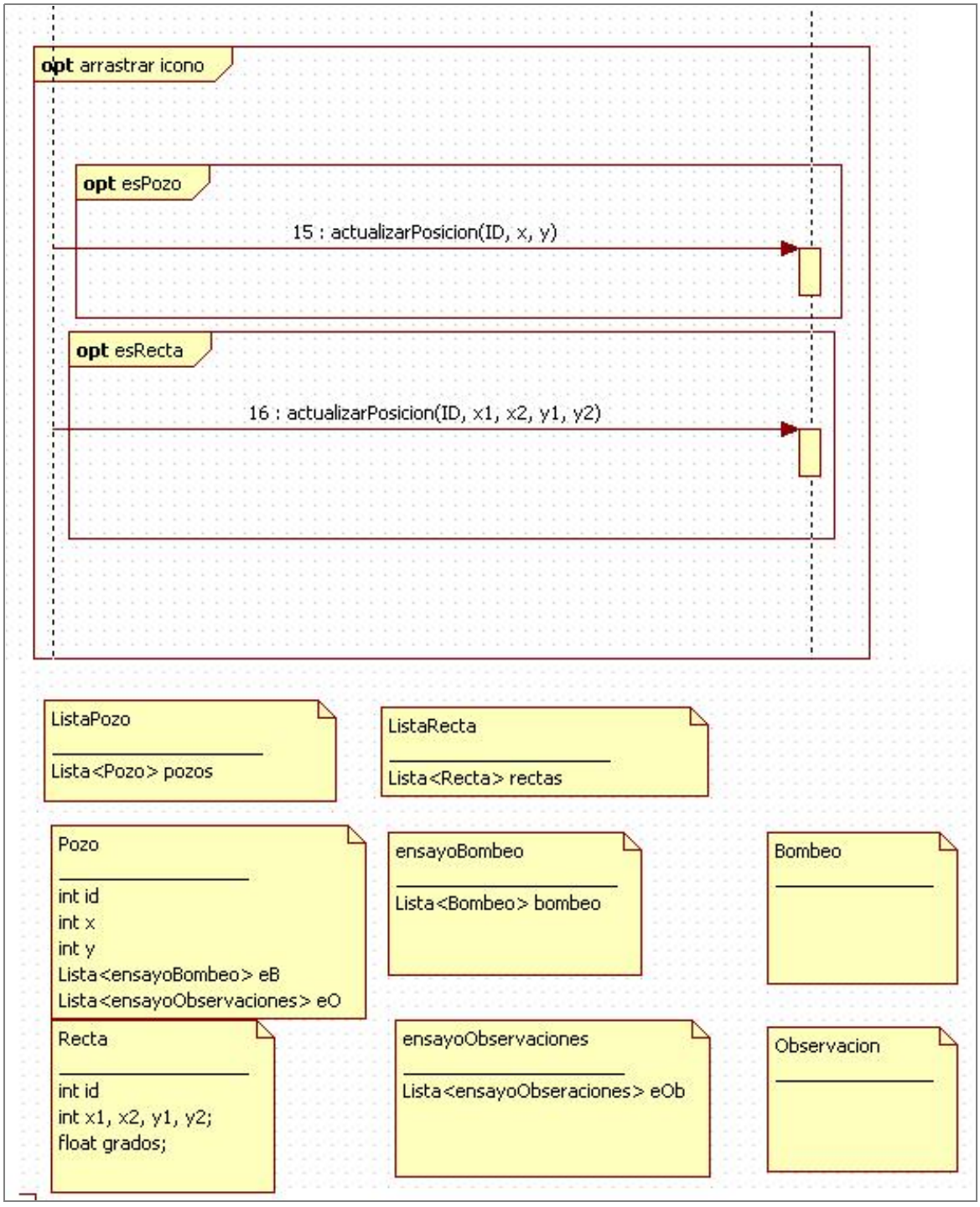
4.6.1 Crear Proyecto



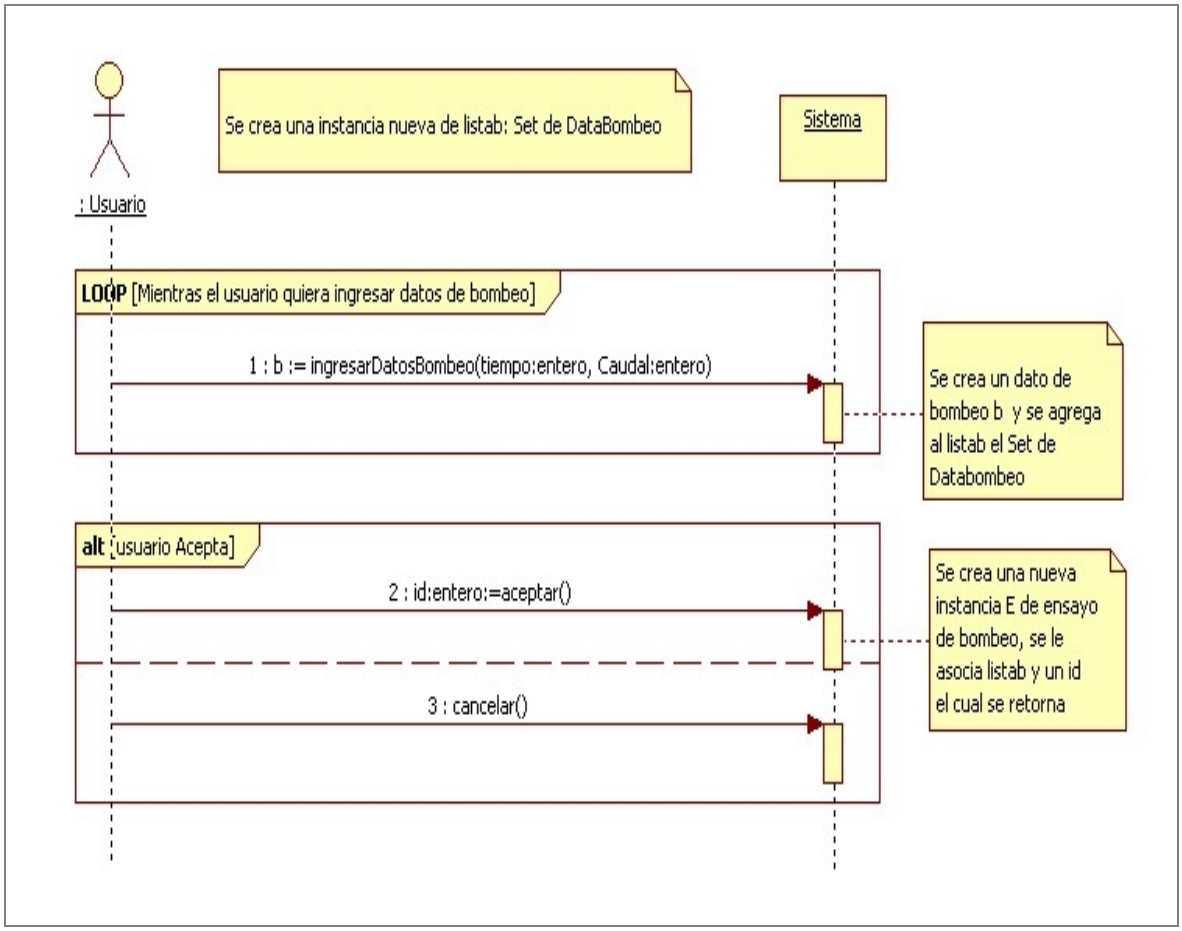
4.6.2 Definir Dominio

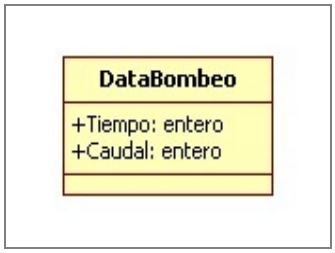



opt listaPozos != null
listaEnsayoCargados:= 7 : asociar(int pozoID)
opt agregarObservacion
listaObservacionesNA:= 8 : agregarObservacion()
Se asocian las observaciones al pozo
y se desasocian los ensayos de
la lista de ensayos sin asociar.
opt asociar
9 : enviarObservaciones(ensayoObservaciones eOb)
alt
10 : cancelarAsociacion()
opt agregarBombeo
listaBombeo:= 11 : agregarBombeo()
Se asocian los ensayos al pozo
y se desasocian los ensayos de
la lista de ensayos de bombeo sin asociar.
seq asociar
12 : enviarEnsayos(ensayoBombeo eBom)
seq asociar
13 : cancelarAsociacion()
14 : cerrar()

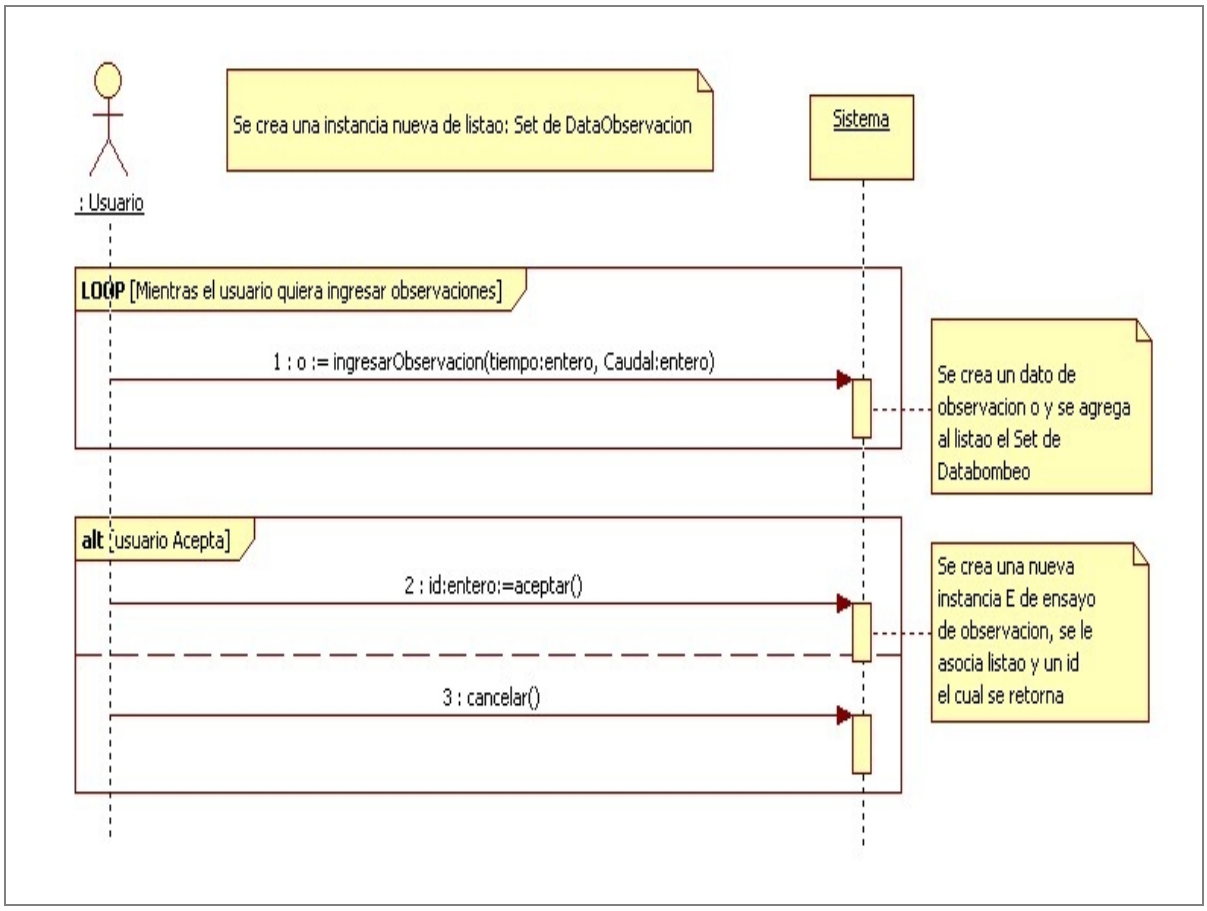

opt arrastrar icono
opt esPozo
15 : actualizarPosicion(ID, x, y)
opt esRecta
16 : actualizarPosicion(ID, x1, x2, y1, y2)
ListaPozo
Lista<Pozo> pozos
ListaRecta
Lista<Recta> rectas
Pozo
int id
int x
int y
Lista<ensayoBombeo> eB
Lista<ensayoObservaciones> eO
ensayoBombeo
Lista<Bombeo> bombeo
Bombeo
Recta
int id
int x1, x2, y1, y2;
float grados;
ensayoObservaciones
Lista<ensayoObseraciones> eOb
Observacion

4.6.3 Ingresar Caudal Bombeado

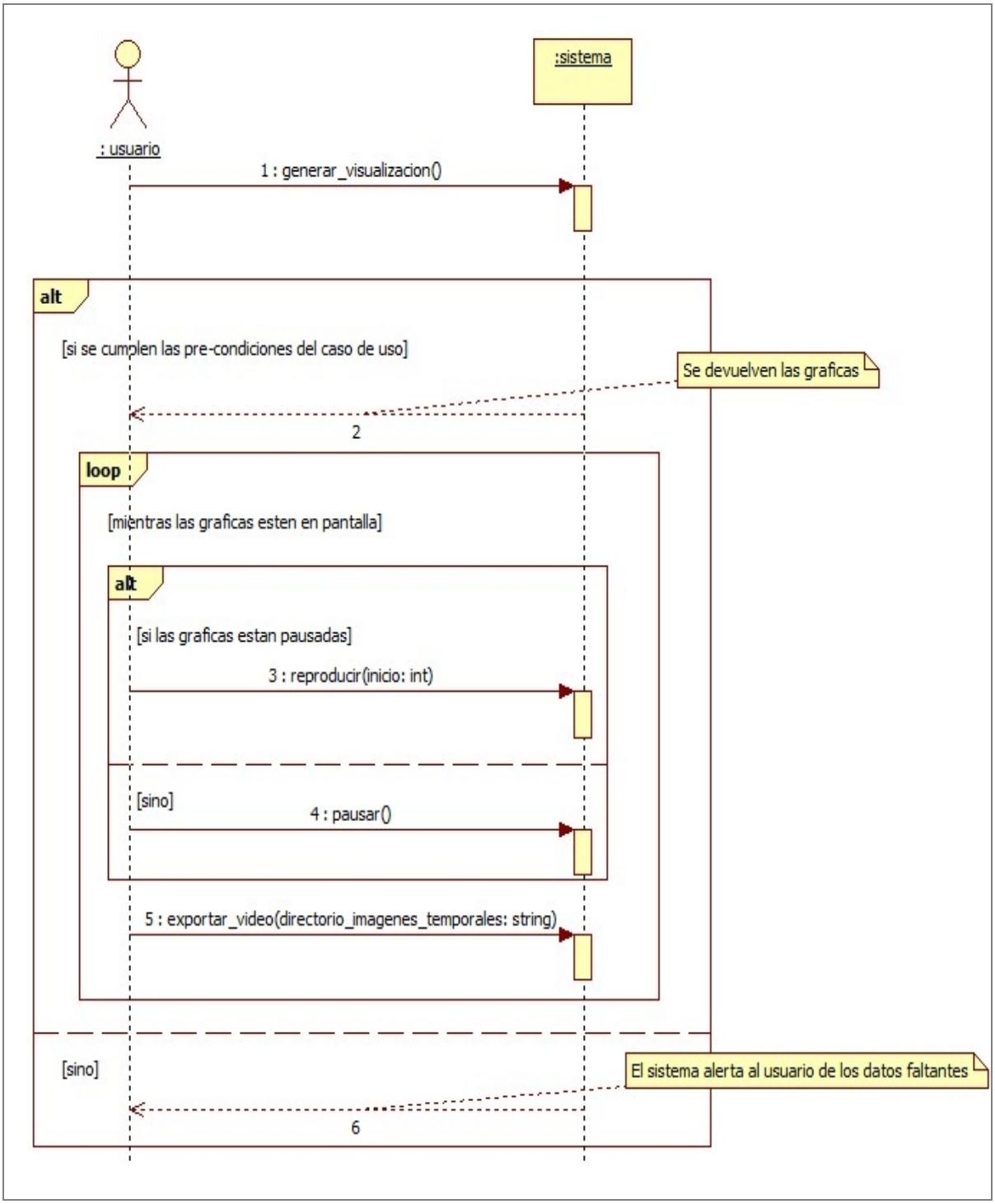




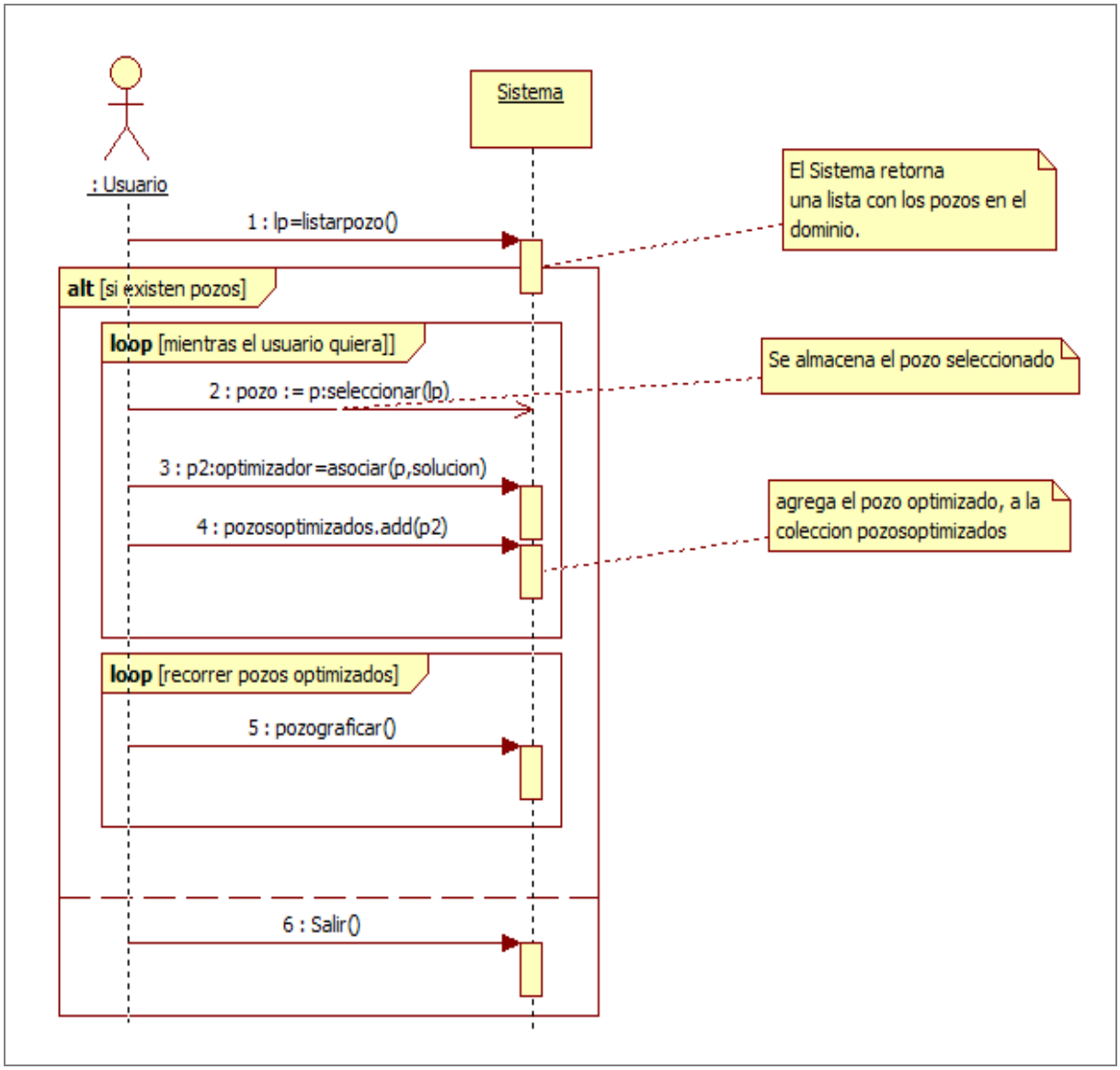
4.6.4 Ingresar observaciones del ensayo



4.6.5 Crear visualización



4.6.6 Selección de Optimización

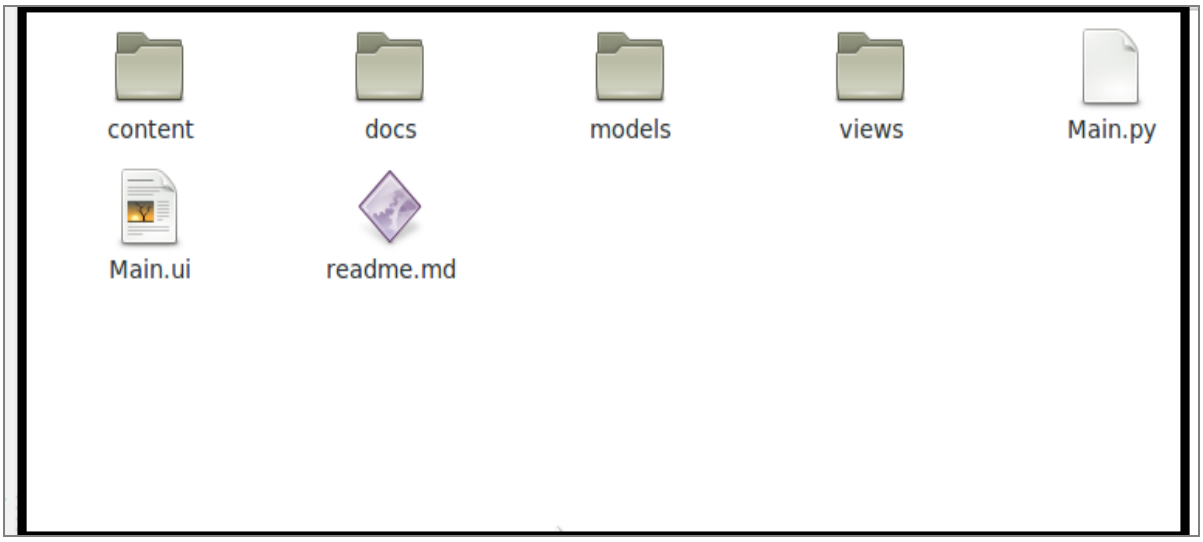


4.7 Estructura del proyecto

La abstracción en un proyecto de esta magnitud, no solo se manifiesta para el análisis o la elección arquitectónica. También conviene establecer convenciones para la organización de los archivos del proyecto. A continuación una descripción de ello.

Estructura del proyecto:

Raíz: Es la carpeta central, en la imagen se representan las carpetas que hacen posible esta abstracción.



content: Carpeta contenedora de los elementos gráficos utilizados por el programa. Ejemplo: Imágenes, modelos de elementos de graficado.

docs: Documentación del proyecto.

models: Se almacenan todas las clases utilizadas.

views: Se guardan las diferentes interfaces gráficas, con las que el usuario interactúa.

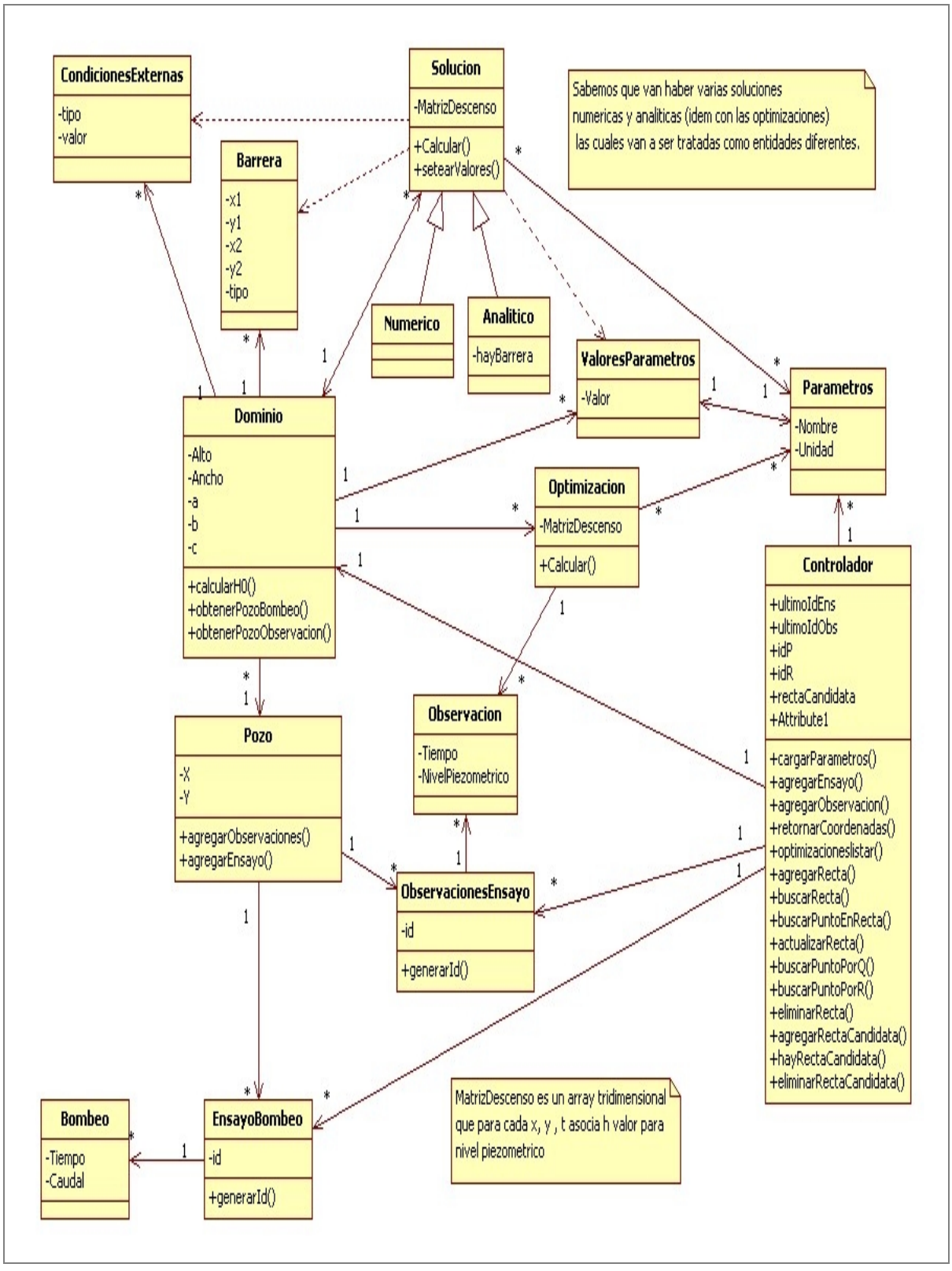
Estas últimas dos carpetas facilitan la abstracción del modelo MVC [1] dado que los modelos, y vistas mantienen una separación entre cada parte.

Main.py: Es el archivo principal de la aplicación; es quien la inicia y por el cual las diferentes vistas y modelos son instanciados en el sistema.

1 Véase la sección 3.1

4.8 Diseño

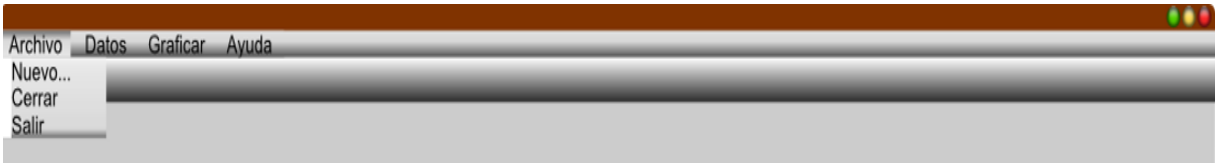
4.8.1 Diagrama de clases



4.8.2 Vistas

4.8.2.1 Menú Pantalla Inicial

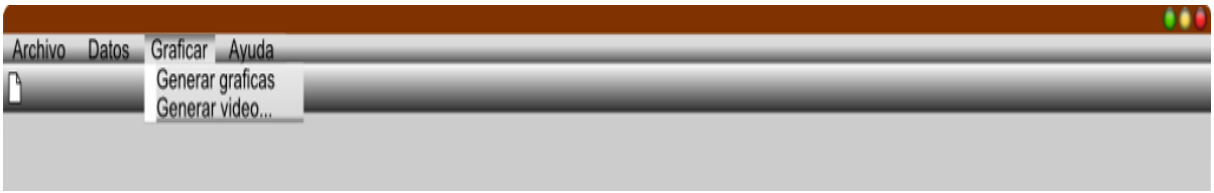
4.8.2.2 Menú Archivo



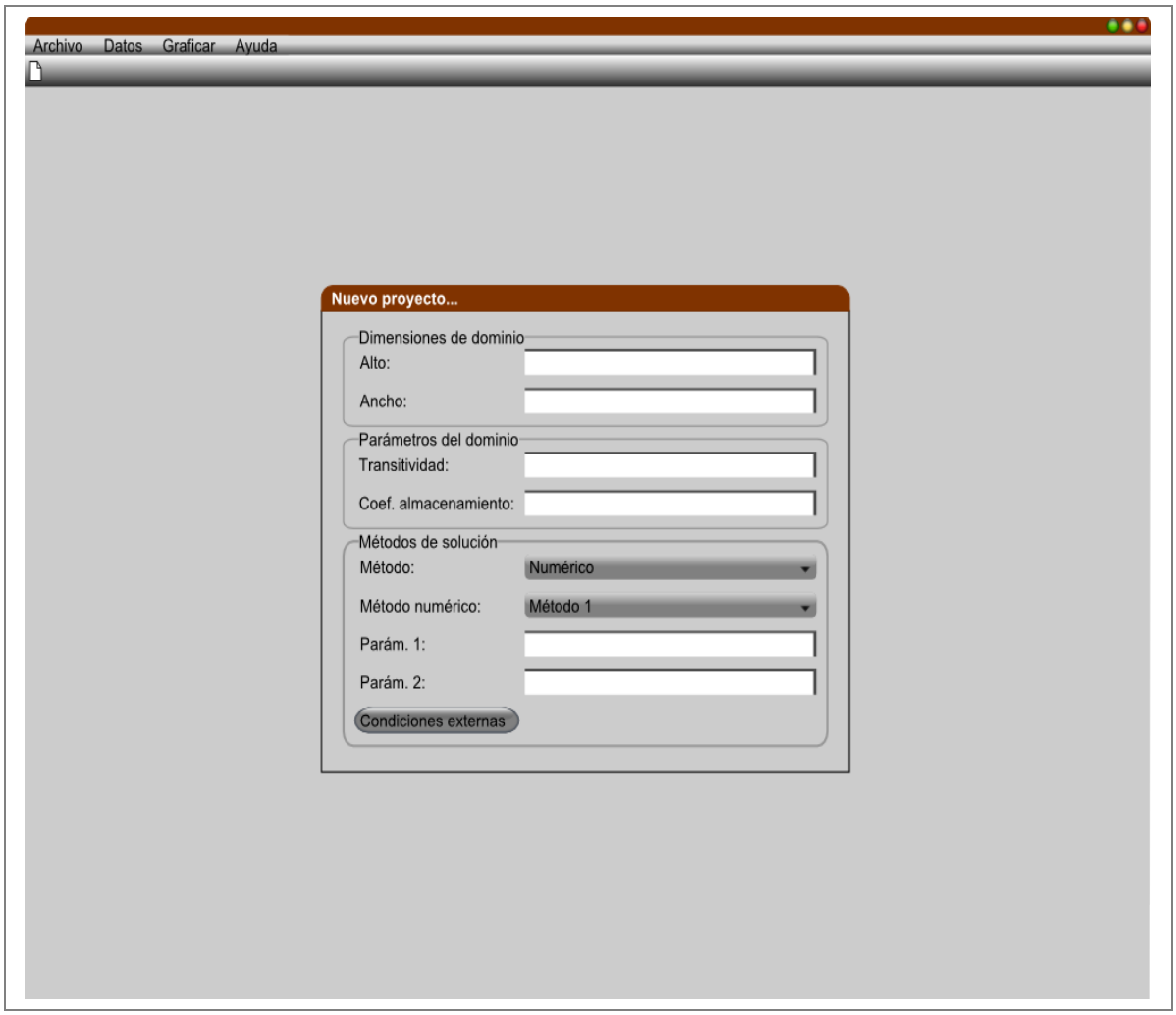
4.8.2.3 Menú Datos

4.8.2.4 Menú Datos observaciones

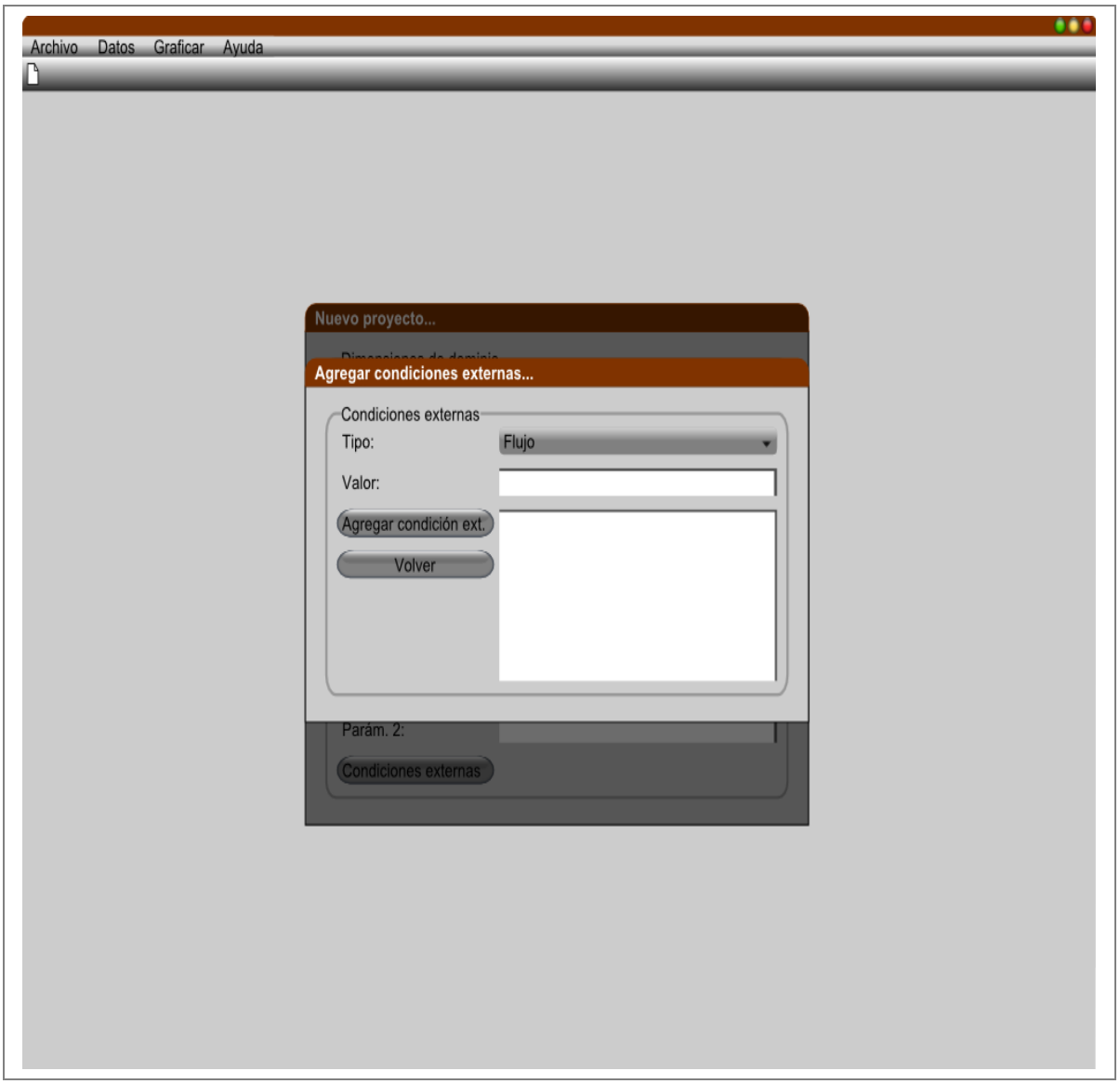
4.8.2.5 Menú Graficar



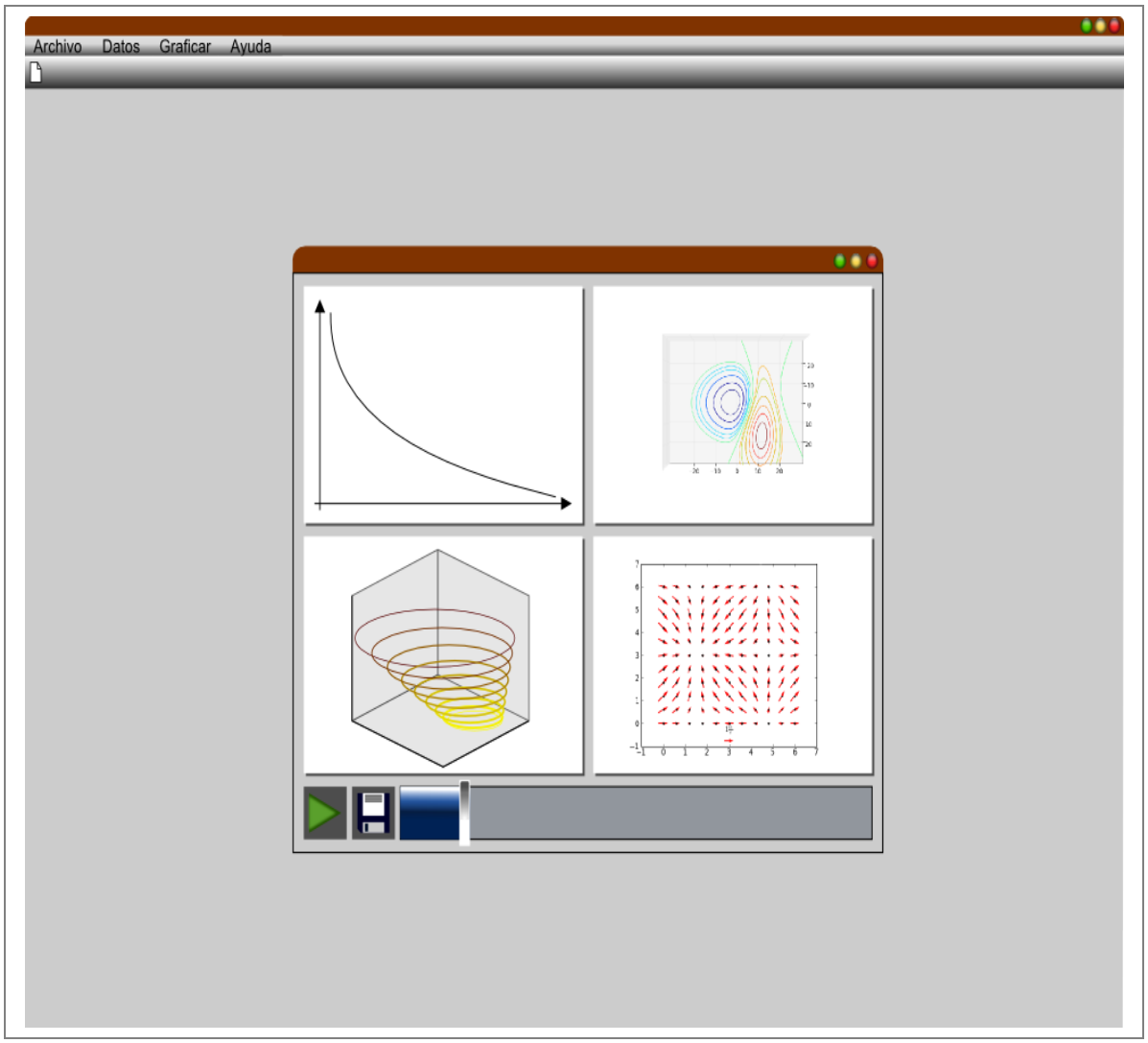
4.8.2.6 Nuevo Proyecto



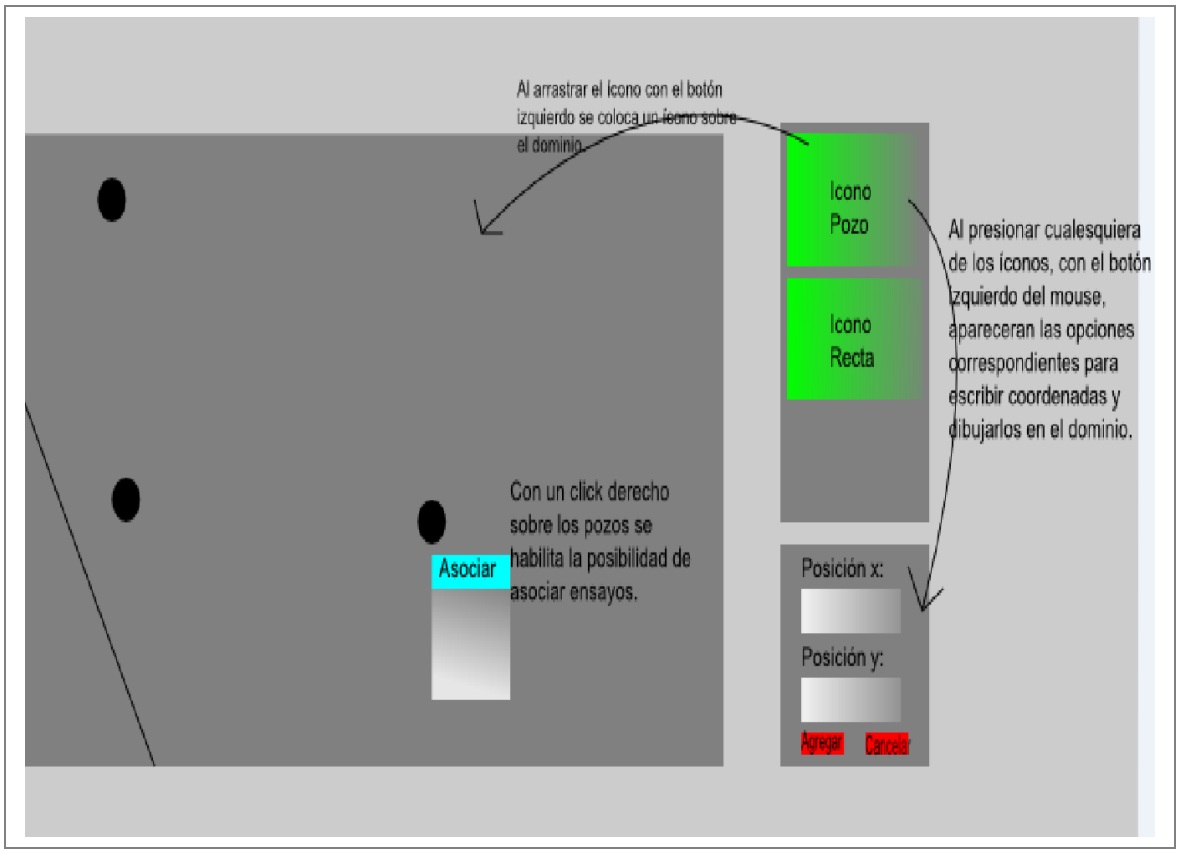
4.8.2.7 Nuevo Proyecto, Condiciones Externas



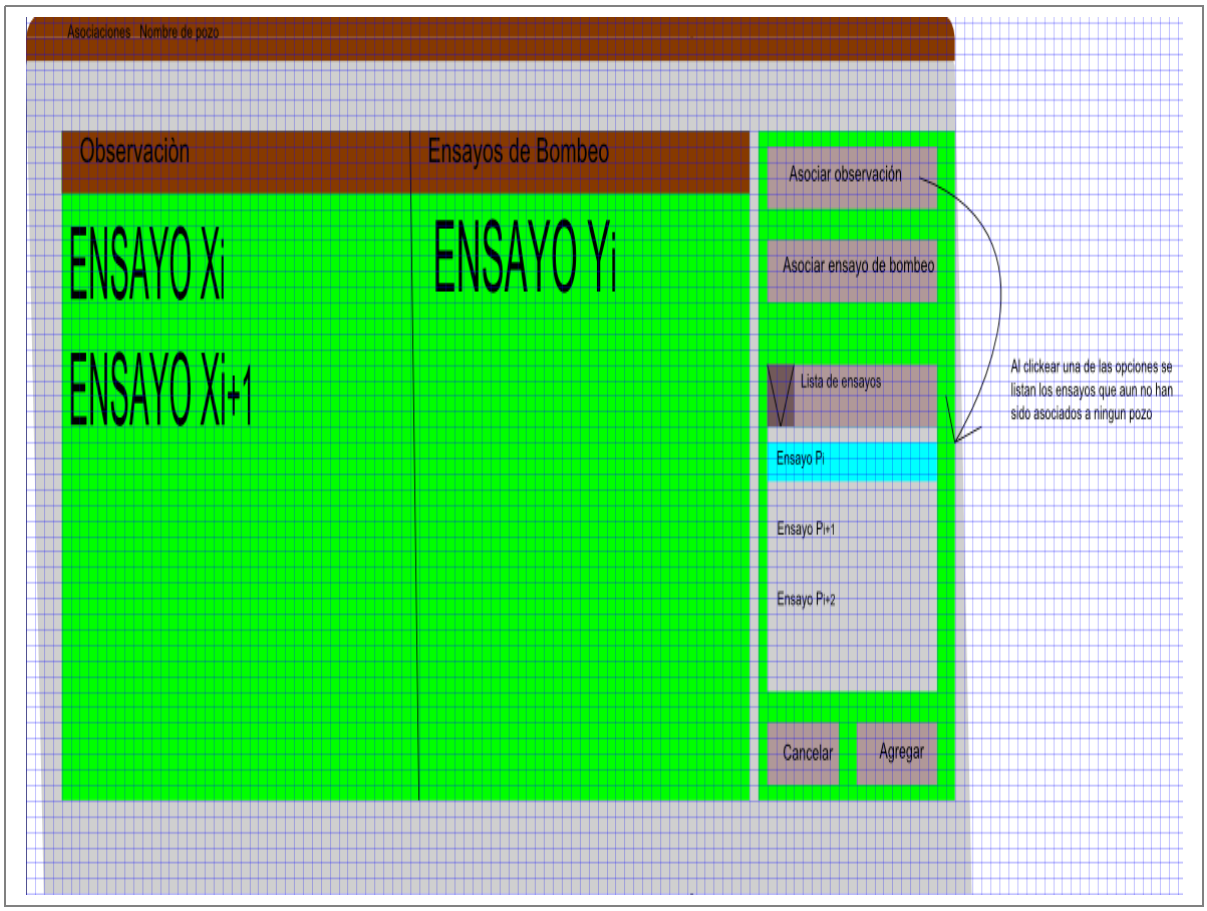
4.8.2.8 Generar Gráficas



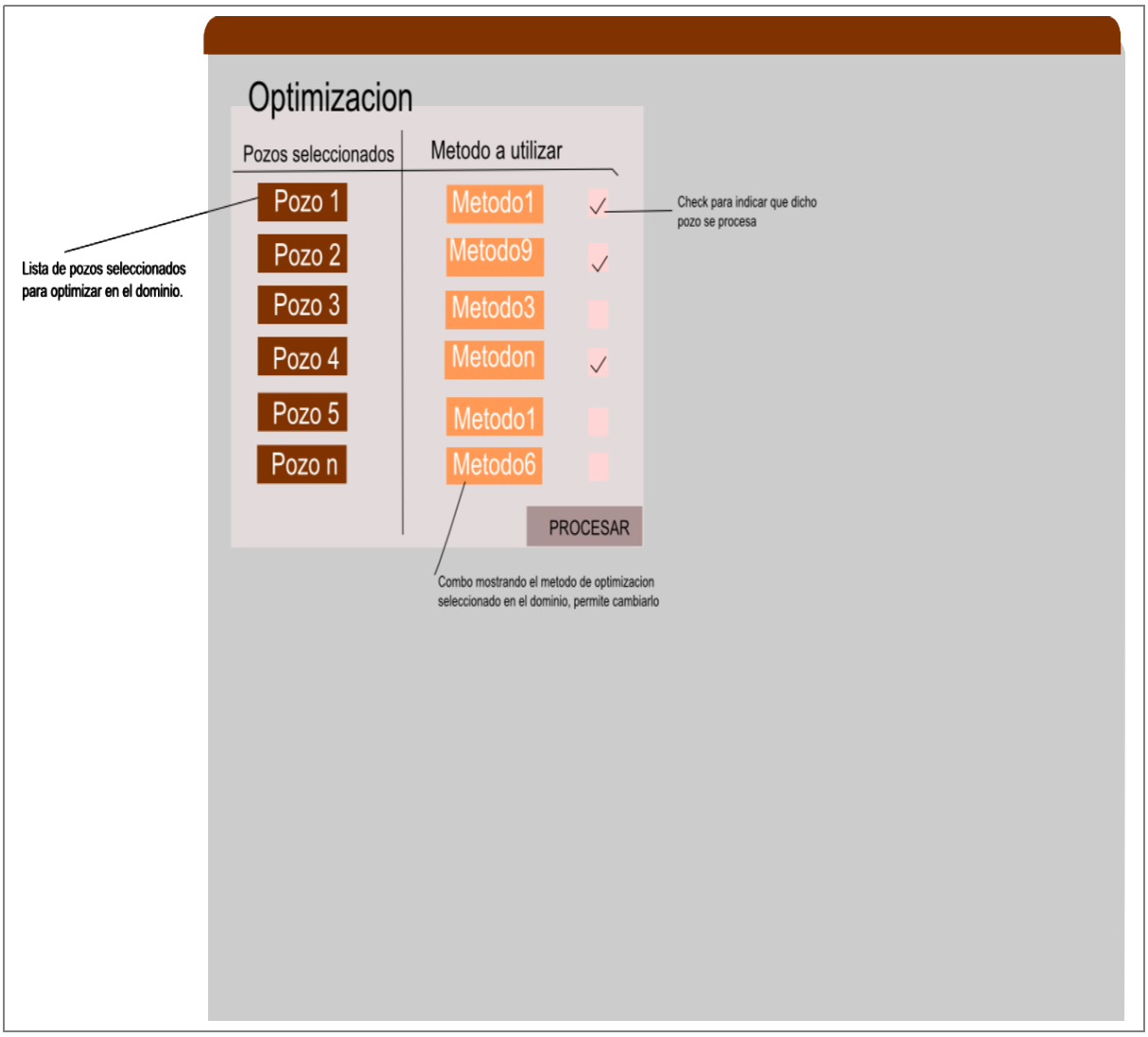
4.8.2.9 Asociar Optimización en el Dominio



4.8.2.10 Ventana de Asociación de Ensayos



4.8.2.11 Asociar Optimización – Confirmar Procesamiento



4.9 Implementación

El lenguaje de programación que decidimos utilizar fue Python. El mismo incluye poderosas herramientas y librerías, conformando una gama DSL [1] que contribuyó a la implementación de la aplicación. No nos decidimos por la utilización de IDE[2] alguno, simplemente aprovechamos las propiedades de editores livianos como notepad ++ [3] o gedit [4]. Para el graficado se empleó Matplotlib, y para operaciones matemáticas diversas la librería numpy[5]. La interfaz de usuario se generó a través de PyQt4[6]. Para el control de versionado creamos un repositorio en github[7], al cual se suben todas las últimas actualizaciones.

4.9.1 Modelo de datos

El modelo de datos, importante para el sistema desarrollado, está basado en un conjunto de clases relacionadas entre sí. Su información es persistida en memoria todo el tiempo que dure el trabajo sobre cada proyecto.

Para el funcionamiento del sistema es necesario contar, en cada caso de uso, con un conjunto de datos cargados. Deben de existir variables globales que afectarán todo el proceso; ellas son T, S, a, b y c. En la creación del proyecto también se definen las condiciones externas, los métodos de solución y las dimensiones del dominio. En la definición del dominio se crean los elementos que habrá dentro del mismo junto con sus coordenadas, pudiéndose ulteriormente asociar datos, en caso de que se hayan cargado, de observación y de bombeo a los pozos, y declarar el signo de la recta que se vaya a crear.

La funcionalidad de graficado requiere que ciertos datos estén cargados en el sistema antes de funcionar. La primer gráfica necesita de una matriz de dos dimensiones donde se ubiquen los niveles piezométricos con su tiempo correspondiente. La segunda gráfica y la tercera necesitan una matriz con el nivel piezométrico del dominio, además del alto y el ancho del mismo.

1 DSL, Domain Solution Language, término que aplica a soluciones para un problema determinado en un lenguaje dado. La solución puede ser la creación de una librería o herramienta.

2 Los IDE'S proveen herramientas que facilitan la programación.

3 Véase la referencia 37 del capítulo “Referencias” del documento del documento “Glosario, apendice y bibliografía”.

4 Véase la referencia 38 del capítulo “Referencias” del documento del documento “Glosario, apendice y bibliografía”.

5 Véase la referencia 27 del capítulo “Referencias” del documento del documento “Glosario, apendice y bibliografía”.

6 Véase la referencia 39 del capítulo “Referencias” del documento del documento “Glosario, apendice y bibliografía”.

7 Véase la referencia 54 del capítulo “Referencias” del documento del documento “Glosario, apendice y bibliografía”.

4.9.1.1 Listado de clases existentes

barrera

Almacena las coordenadas, y el signo de la recta creada en el dominio.
Posee un método de actualización, y del cálculo de una recta que pasa entre dos puntos, esto para extender sus dimensiones en el plano.

bombeo

Contiene los datos del tiempo y el caudal registrados para un bombeo individual. Posee un método datosNombre que permite recuperar la lista de atributos de esta clase y un método devolverAt que permite recuperar el valor que contiene cada uno de sus atributos a través de un índice. Ambos métodos son utilizados por la clase modelotabla la cual permite implementar el patrón Modelo Vista de PyQT.

calitheis2

Clase correspondiente al método de optimización (válido solamente para el método de solución analítica de Theis) del mismo nombre; la misma hereda de la clase "metodooptimizacion", la cual en el método calcular se desarrolla el algoritmo para procesar esta optimización.

condicionExterna

Contiene dos atributos uno Tipo y otro Valor. Una condición externa o de contorno, se definirá solo para las soluciones numéricas no para las analíticas; para cada contorno se podrá definir 2 tipos de condiciones de contorno: de nivel o de flujo.

controlador

Para esta aplicación se utilizó una única clase controladora del tipo Façade debido a que se debe implementar un número reducido de funcionalidades las cuales dependen unas de otras, ya que para la ejecución de una funcionalidad se exige que previamente el usuario haya ejecutado otra serie de pasos anteriores. Centraliza la manipulación de los diferentes objetos manipulados en el sistema: dominio, parámetros, ensayos, observaciones, pozos, puntos, rectas, optimizaciones.

dominio

Contiene las dimensiones del área a trabajar, además de los parámetros a, b y c. Puede contener objetos tipo: Barrera, CondicionExterna, Pozo, Optimización y MetodoSolucion.

ensayobombeo

Contiene la lista de bombeos y el identificador numérico para cada ensayo de bombeo. Al igual que la clase bombeo, contiene los métodos datosNombre y devolverAt que facilitan la implementación del patrón Modelo Vista de PyQT.

figura

Esta clase contiene las gráficas que luego se dibujarán en pantalla. Aquí se ingresan algunos datos y se calculan las gráficas.

metodooptimizacion

Clase base para todos los algoritmos de optimización existentes en el sistema. Contiene la lista de atributos comunes a los mismos, tales como lista de parámetros, matriz de descenso, lista de observaciones. Asimismo contiene un método encargado de procesar los datos aplicando el algoritmo de optimización y así generar resultados a ser mostrados en gráficas. Cada método de optimización existente heredará de esta clase.

metodoSolucion

Clase base para todos los métodos de solución existentes en el sistema. Contiene una referencia al dominio y una lista de parámetros. Existen dos clases que heredan de esta: son metodoAnalitico y metodoNumerico. Posee un método setearValores para setear los valores utilizados por cada parámetro y un método calcular para efectuar el cálculo de las variaciones de los niveles piezométricos sobre todo el dominio. El cálculo puntual para un punto del dominio es implementado por un método de cada instancia particular.

observacion

Contiene los datos del tiempo y el nivel piezométrico registrados para una observación. Posee un método datosNombre que permite recuperar la lista de atributos de esta clase y un método devolverAt que permite recuperar el valor que contiene cada uno de sus atributos a través de un índice. Ambos métodos son utilizados por la clase modelotabla la cual permite implementar el patrón Modelo Vista de PyQT.

observacionesEnsayo

Contiene la lista de observaciones y el identificador numérico para cada conjunto de observaciones. Al igual que la clase Observacion, contiene los métodos datosNombre y devolverAt que facilitan la implementación del patrón Modelo Vista de PyQT.

parametros

Contiene el nombre y la unidad de cada parámetro utilizados en el dominio y puede contener también el valor asociado al mismo. Esta clase es utilizada por el controlador para hardcordear los parámetros que se podrán utilizar en el sistema.

pozo

Contiene las coordenadas para cada pozo del dominio.
Sus métodos permiten actualizar coordenadas, agregar ensayos y observaciones.

t**heis**

Es una clase que hereda de metodoAnalitico que a su vez hereda de metodoSolucion. Contiene los métodos calcularpozo y WTheis utilizados para la aplicación puntual del método a Theis a un punto dentro del dominio.

4.9.2 Implementación de cálculos

4.9.2.1 Niveles inciales

El agua en un acuífero se mantendrá en un determinado nivel fijo conocido antes de que se efectúe el correspondiente ensayo de bombeo. En el cálculo de los diferentes métodos de solución, se considera como datos conocidos los niveles de agua iniciales en todos los puntos del dominio.

Por esto, definimos el nivel inicial del acuífero en un punto como Ho. Este valor queda determinado por la evaluación en las coordenadas del punto de la siguiente ecuación del plano:

$$Ho = a * x + b * y + c$$

donde "x" e "y" son las variables independientes que serán reemplazadas por las coordenadas del punto y sus coeficientes "a","b" y "c" son unas constantes.

Para implementar esto en nuestro sistema, recurrimos a agregar tres nuevos atributos a la clase dominio: a ,b,c. Estos atributos son del tipo real (float en Phyton). Los valores para estos atributos serán ingresados en el momento de crear un nuevo proyecto y determinarán los coeficientes del plano de la ecuación mencionada.

Luego de la aplicación de un método solución se obtiene un valor que determina el descenso “s” en un punto y tiempo determinado. Sin embargo, este no es igual al nivel “h”. Este se calcula como “h=Ho-s”.

Por lo tanto, el nivel en un tiempo “t” y en un punto de coordenadas "xq" e "yq" se calcula como Hoq= a*xq+b*yq+c -s1(xq,yq,t)

En nuestro sistema, esto fue implementado de la siguiente manera. Por un lado, se invoca al método “calcularpozo” de la Solución correspondiente (Theis, Hantush) que se encarga de efectuar el calculo del descenso puntual. Por otro lado, se invoca al metodo devolverH0 de la clase dominio que toma como parámetros las coordenadas (x,y) del punto. Finalmente, se opera para llegar al resultado.

4.9.2.2 Postprocesos

Tras observar los scripts de Matlab enviados por el grupo de la Regional Norte, mas explicaciones efectuadas, pudimos darnos cuenta de la importancia y el fin de utilizar postprocesos en la aplicación para garantizar la correcta interpretación de los ensayos de bombeo en nuestro sistema. Estos se deben básicamente al principio de superposición de efectos, el cual permite sumar los efectos de los diferentes fenómenos.

Supongamos que tenemos un pozo que en el tiempo 0 comienza bombeando un caudal de 480, hasta detenerse en el tiempo 0.18 (cuadal 0). Para simular este hecho, lo que considera nuestro tutor y que se ve reflejado en sus scipts, es agregar un pozo virtual en la misma posición y que comienza a bombear en t=0.18. De esta manera el caudal extraído para este pozo para t>0.18 es Q= 480 + -480 =0.
Para implementar esto, en nuestro sistema, consideramos que era innecesaria la creación de tantos pozos virtuales como caudales de bombeo tuviera asociado un pozo de bombeo. Mas teniendo en cuenta que estos pozos conservaban las mismas coordenadas que las del pozo original, que el posproceso se efectuaba de forma independiente para cada pozo y que la incorporación de nuevos pozos de forma repentina nos podría generar complicaciones al momento de manipular sus elementos gráficos vinculados en el espacio de trabajo.

Por lo tanto, para dar solución a esto se agregaron a la clase “ensayobombeo”, un nuevo atributo “bombeosproc” y un nuevo método “proceso” el cual fue innovado, para procesarse en el momento en que se instancia un objeto de esta clase garantizando de modificar la lista de bombeo de forma adecuada para respetar el principio mencionado. Luego de que la lista de bombeo es cargada en el sistema, se podrá acceder tanto a sus valores originales si fuera necesario (visualización de los datos), o a sus valores procesados para efectuar cálculos con éstos.

Para el caso de la interacción con barreras, sí fue necesario utilizar pozos virtuales, los cuales son manipulados de forma independiente a los pozos reales, a través del seudo atributo “pozosVirtuales” de la clase “dominio”. También se definió un método “procesarBarrera” dentro de la misma clase. Por el mismo motivo, se incluyen tres atributos: “alfa”, “beta” y “gamma” para el seteo de los coeficientes de la recta en el momento que es instanciada la barrera y que luego son utilizados en el postproceso. La implementación de este postproceso se basó en gran medida en los script enviados, adaptándolos por supuesto, a la programación orientada a objetos.

4.9.2.3 Librerías matemáticas

Para la implementación de la mayor parte de las funciones matemáticas se utilizó la librería **Numpy** [1]**,** para el graficado se utilizó la librería **Matplotlib**[2]. Con ambas, se abarcan casi todas las funciones utilizadas en los script enviados y que son provistas por Matlab[3].

Sin embargo, para codificar el método Hantush, tuvimos que recurrir a un par de funciones matemáticas un poco mas complejas. Es por esto que recurrimos a la librería **Scipy**[4] la cual originalmente pensábamos utilizar para interpolación bilineal[5], lo que no fue necesario. El módulo “special” de esta librería nos permitió implementar una integral de una exponencial[6] y la función Bessel modificada de

1 Véase la referencia 52 del capítulo “Referencias” del documento “Glosario, apéndice y bibliografía” para conocer mas sobre las funciones de Matlab y sus equivalentes en Numpy
2 Véase el punto 2.3.1.5 del Apéndice.
3 Véase capítulo sobre Herramientas científicas existentes, en Estado del Arte.
4 Véase la referencia 27 del capítulo “Referencias” del documento “Glosario, apéndice y bibliografía”.
5 En tratamiento digital de imágenes, procediminiento de interpolación basado en promediar los valores de los 4 píxeles vecinos.
6 Véase la referencia 55 del capítulo “Referencias” del documento “Glosario, apéndice y bibliografía” para una introducción matemática a la integral exponencial.

tercer grado[1]. Así logramos reemplazar las funciones “expint” y “besselk” provistas por Matlab, por “expn” y “kn” respectivamente.

Todas las funciones matemáticas implementadas fueron evaluadas en módulos de prueba para verificar resultados idénticos en los ambientes mencionados.
Cabe aclarar, algo que no es menor pero que puede resultar en sutileza, es el cálculo de gradiente[2], que se consigue con una función del mismo nombre pero aplicada a una matriz de tres dimensiones, con Numpy primero retorna el gradiente en y y luego en la variable x, sin embargo, en Matlab sucede lo contrario.

4.9.3 Interfaz

Para la presentación de la interfaz de usuario hicimos uso de la herramientas PyQt4 [3]. La misma cuenta con un conjunto de clases de alto y bajo nivel, divididas en módulos [4]. Los que hemos utilizado son los siguientes:

QtCore: Módulo para tareas de bajo nivel; todos los restantes módulos se basan en QtCore. Es quien lleva a cabo las funcionalidades no gráficas.

QtGui: Módulo para tareas de alto nivel; conjunto de clases utilizadas para la presentación de los datos al usuario.

Se ha hecho uso de otros módulos de Python:

Sys: Este módulo provee acceso a algunas variables usadas o mantenidas por el intérprete y funciones que interactúan fuertemente con el intérprete.

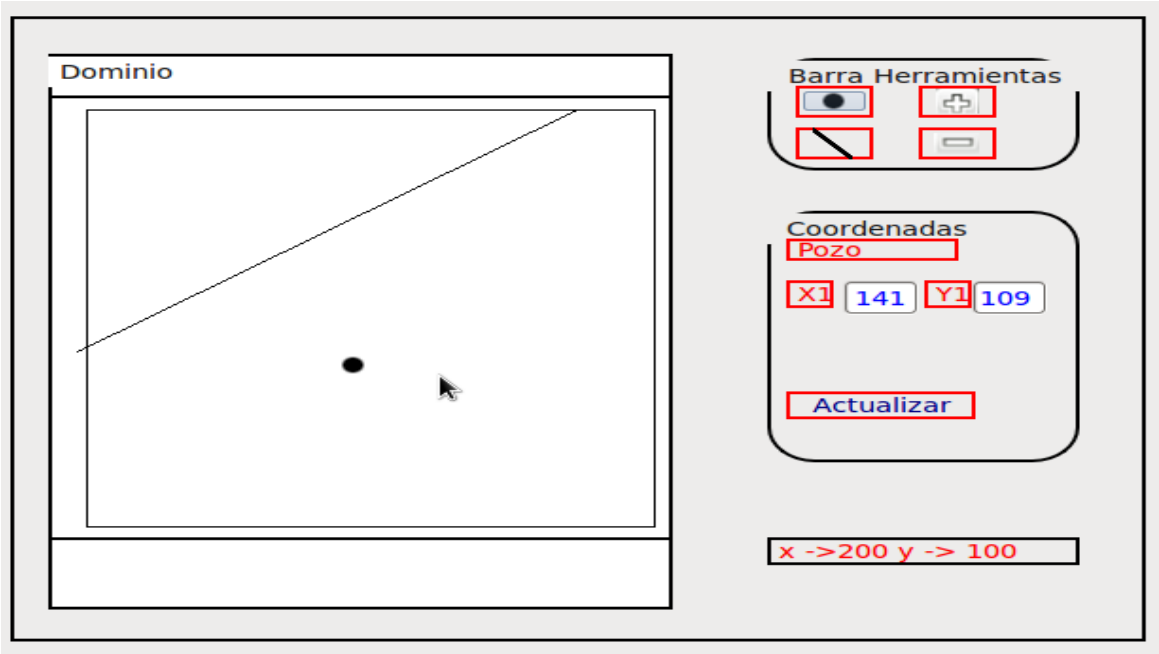
Zipfile: Módulo para leer y descomprimir archivos comprimidos.

Xml.dom.minidom: Es una implementación liviana de la interface DOM (Document Object Model) utilizada para parsear documentos XML como estructuras de árbol.

4.9.3.1 Definir Dominio

Cuenta con tres áreas bien definidas; ellas son barra de herramientas, área de coordenadas y plano.

1 Véase la referencia 56 del capítulo “Referencias” del documento “Glosario, apéndice y bibliografía”.
2 Véase la referencia 53 del capítulo “Referencias” del documento “Glosario, apéndice y bibliografía” para saber que resultado devuelve la aplicación de la función gradient en Matlab.
3 Véase “PyQt4” en la sección “Pyhton, Herramientas: Librerías, Frameworks, Ámbito de desarrollo” del capítulo “Evaluación de lenguajes” del documento “Estado del Arte”
4 Véase la referencia 39 del capítulo “Referencias” del documento del documento “Glosario, apendice y bibliografía”



Los botones de la barra de herramientas que identifican a barreras y pozos heredan de la clase boton, la cual lo hace de QtGui.QPushButton. Esto debido a la necesidad de reimplementar los métodos: mousePressEvent y mouseMoveEvent, con los cuales se determina si el usuario va a utilizar el botón para iniciar una interacción con el dominio, o si va a ingresar el elemento mediante coordenadas, mientras que los botones para efectuar las tareas de ampliación y reducción son instancias directas de QtGui.QPushButton.

El área de coordenadas es instancia de una clase llamada gbCoordenadas, la cual hereda de QtGui.GroupBox. Esta clase performa las acciones de actualizar las coordenadas en determinados eventos como cuando el mouse presiona en el dominio algún pozo o barrera, cuando se crea algún elemento o si éste es soltado.

Para la representación del dominio se ha utilizado un framework de Qt, QGraphics Framework, el cual tiene como fin implementar gráficas utilizando un modelo MVC, y orientado a objetos[1].

El dominio instancia la clase vistaGrafica, la cual hereda de QtGui.QGraphicsView. Es la clase más compleja de la definición del dominio ya que se sobreescriben métodos para permitir el drag and drop [2] de objetos sobre ella. También se debe saber qué acciones realiza el mouse dentro del dominio, por lo cual se deben reimplementar métodos como: mouseMoveEvent, mousePressEvent y mouseReleaseEvent.

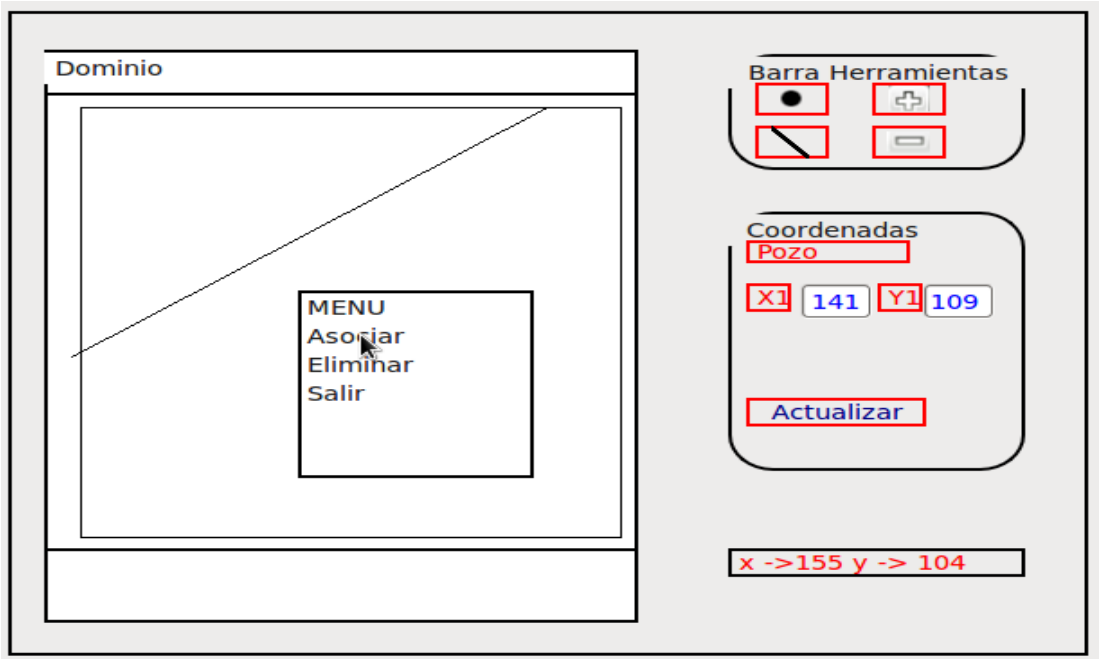
Los elementos fundamentales para representar en el dominio son instanciados a través de dos clases bien definidas: Para los pozos, vistaPozo, y para las barreras, vistaBarrera, la primera hereda de QtGui.QGraphicsPixmapItem, y la segunda de QGraphicsLineItem.

1 Para una introducción al framework, consúltese apéndice 2.3.1.5 10.1 Graphics View Framework, para un mayor detalle consulte la referencia 57.
2 Véase la definición de Drag and Drop en el capítulo “Glosario” del documento del documento “Glosario, apéndice y bibliografía” .

Estos elementos forman parte del modelo del framework, y heredan de dos clases específicas para la representación de imágenes y líneas.

La comunicación entre el modelo, junto con su organización en la gráfica, y la vista es realizada por un intermediario, llamado escena, el cual es independiente de la vista en la cual se muestra, dándole esto último en un sentido abstracto verdadera forma dentro del modelo MVC.

Existe una clase menú, que podría considerarse como una cuarta área, llamada menu, la cual hereda de QtGui.QListView quien se encarga de implementar la clase abstracta QabstracListView. Lo que hace es proveer una vista para un conjunto de modelos [1], representándolos en forma alistada. Pragmáticamente menú presenta una serie de opciones que se pueden efectuar sobre los elementos del dominio.



(Menú actuando sobre un punto del dominio)

Existe dentro de la vista de definir dominio una clase global, la cual almacena variables de estado, que permite la comunicación entre las cuatro áreas definidas hasta el momento. Su nombre es elementoDominio.

Las variables de estado son las siguientes:

elementoDominio = [0 ! 1]

Es una bandera de control, que le dice a la aplicación qué tipo de elemento se está arrastrando al dominio para ser creado. 0 es un pozo, 1 es una barrera.

reloj = [Bool]

transicion = [Bool]

1 QAbstractListView, y QListView son clases que forman parte del modelo MVC de Qt. Véase la referencia 40 del capítulo “Referencias” del documento del documento “Glosario, apéndice y bibliografía”

Son banderas de control que le dicen a la aplicación, dependiendo de sus estados, cuándo se puede comenzar a arrastrar un botón de la barra de herramientas.

El estado inicial de estos elementos en cada botón es:

reloj = False

transicion = False

Para cada botón, estas banderas comenzarán a cambiar en el momento en que se presione el botón.

menuMouse = ""

Es una instancia global de la clase menu(QtGui.QListView)

En su función de inicio se setean las acciones estáticas, como salir o eliminar.

selectedMenuMouse = {}

Contiene una referencia al tipo de elemento en el dominio sobre el cual el usuario hubo aplicado un click derecho.

selectedMenuMouse es un diccionario que contiene las claves/valor:

Tipo = ['Punto' | 'Barrera']

id = Identificador del elemento

Esto permite identificar al elemento del dominio sobre el cual se deben de efectuar las acciones del menú desplegado.

gbCoord = ""
Es una instancia global de la clase gbCoordenadas(QtGui.QGroupBox)

Dominio = ""
Es una instancia global de la clase box(QtGui.QGroupBox)

hayPozoCandidato = [Bool]

pozoCandidato = [int]
El pozo candidato es aquel que es creado mediante coordenadas. Al momento de posicionar este pozo candidato se activa la bandera hayPozoCandidato a verdadera, al tiempo que en la variable pozoCandidato se almacena su identificador, que va a ser utilizado para agregarlo a la lista de pozos, en caso de que el usuario lo acepte.

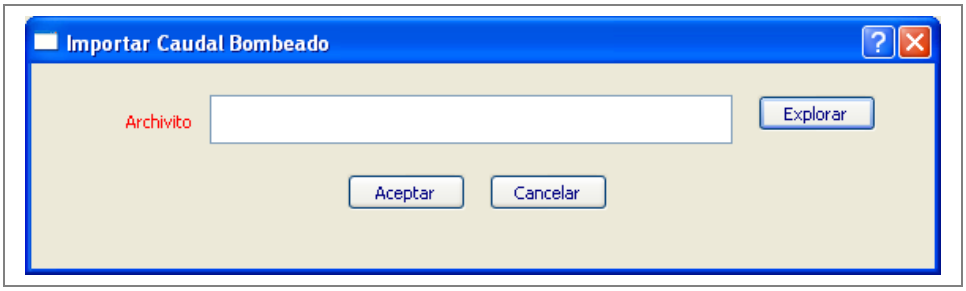
PozoSeleccionado = [int]
Guarda el identificador de un pozo cuando es seleccionado en el dominio.

En caso de no existir pozo seleccionado, su valor es 0.

4.9.3.2 Importar Ensayo de bombeo y observaciones

Al igual que sucede con las demás vistas de la aplicación, se utilizan las clases: QtCore, QtGui de PyQt y la libreria sys de Python.

Para importar ensayos y observaciones se diseñaron simples ventanas que heredan de QtGui.QDialog



Para esta ventana se utilizaron especialmente los diálogos estándar de Qt, los cuales son clases que se heredan directamente de QtGui: **QMessageBox** , para mostrarle mensajes al usuario y **QFileDialog**, para mostrarle un mini explorador de ficheros al usuario para que este navegue dentro de los directorios del sistema de archivos de su computadora y seleccione el fichero que contiene los datos a importar.

Para el caso del procesamiento de los datos contenidos en un fichero de tipo ASCII, se instancia un objeto de la clase **FILE** a través del método open de esta clase y se lee todo su contenido a través del método readlines para que luego éste sea procesado como cadenas de texto.

Ficheros ODF

A través de Pyhton es posible extraer, parsear y obtener contenido de ficheros ODF [1]. El formato **ODF** (Open Document Format) fue diseñado para compartir información a través de diferentes aplicaciones de procesamiento de texto. Un archivo ODF puede tener diferentes extensiones, lo cual determina el tipo de información que almacena. El tipo de archivo que interesa en nuestra aplicación es el ods (OpenDocument Spreadsheet).

Un archivo ODF es básicamente un archivo comprimido, que está compuesto por varios archivos XML. El proceso de lectura del mismo, implica primero descomprimir este fichero y luego acceder al fichero content.xml [2], el cual contiene todos los datos del archivo ODF en sí. Pyhton tiene soporte propio para el manejo de archivos comprimidos a través de la clase **Zipfile**.

El contenido dentro de un archivo se manipula de una forma más adecuada utilizando una estructura de árbol. A través de Pyhton es posible parsear todo un archivo xml y obtener una estructura de árbol. Existen dos tipos de parseos XML: **SAX** y **DOM**. El parseo SAX es más rápido pero permite manipular una etiqueta a la vez e implica un mayor trabajo de programación; en cambio, en el parseo DOM se lee primero todo el archivo en la memoria y esto provee mejores opciones para la navegación y manipulación de los nodos XML.

Los paquetes **xml.dom** y **xml.dom.minidom** de Python permiten leer ficheros XML como árboles DOM. Ambos paquetes vienen con toda instalación estándar de Pyhton. Para este proyecto es suficiente utilizar el paquete xml.dom.minidom que ocupa menos espacio en memoria y es más fácil de usar que el paquete dom.

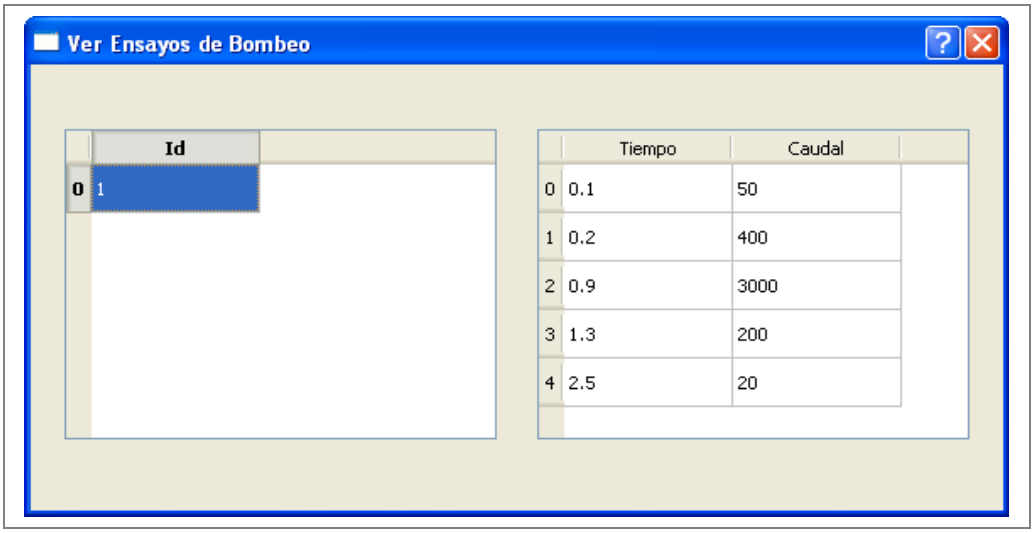
Para el caso del procesamiento de los datos contenidos en un fichero de Open Office Calc (ods), en nuestra aplicación se utilizaron los paquetes Zipfile y xml.dom.minidom**.**

1 Véase la referencia 42 del capítulo “Referencias” del documento del documento “Glosario, apéndice y bibliografía” para tener más información sobre como extraer y parsear archivos ODF en Pyhton.

2 Véase la referencia 43 del capítulo “Referencias” del documento del documento “Glosario, apendice y bibliografía” para leer la documentación de las etiquetas contenidas en el archivo content.xml.

4.9.3.3 Ver Ensayos y Observaciones

Estas ventanas están separadas en dos pero tienen exactamente el mismo funcionamiento. Cada ventana cuenta con dos tablas: **QtGui.QTableView**.

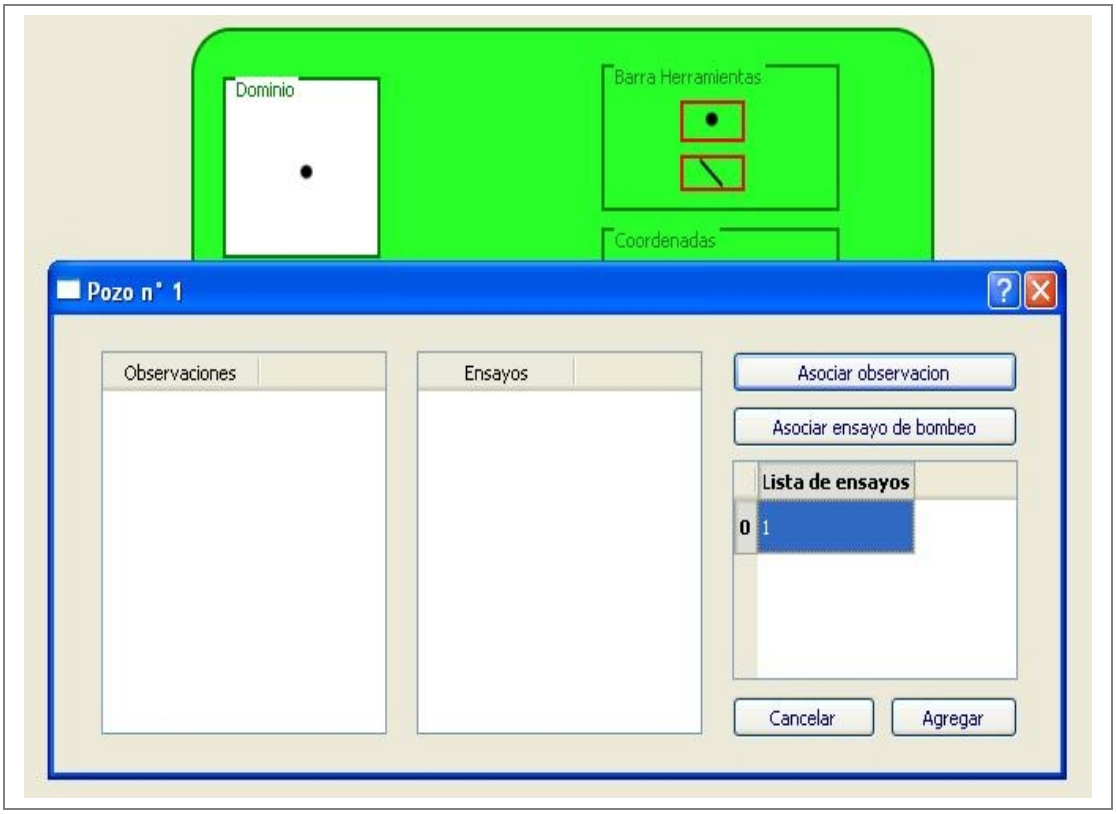


Ambas a pesar de recibir datos diferentes, su modelo (model) es seteado tras instanciar un objeto de la clase modelotabla. La clase modelotabla hereda de la clase **QtCore.QAbstractTableModel,** que a su vez hereda de **QtCore.QAbstractItemModel** y fue incluida en nuestra aplicación con el ánimo de conseguir la aplicación del patrón Modelo Vista de PyQt en la misma.

Esta pretende ser una clase genérica e independiente de los datos que permita desplegar instancias de objetos de las clases del modelo de datos en las diferentes vistas de la aplicación. Esto se logra enviando en el momento de su instanciación, la lista de objetos y la lista de atributos a mostrar, con lo cual puede ser utilizada tanto para listar los ensayos de bombeo o conjuntos de observaciones cargadas en el sistema como para mostrar varias observaciones o varios bombeos. Esta clase implementa los métodos: *headerData, rowCount, columnCount, data* y un método *objeto* para recuperar el dato específico asociado a una determinada celda.

4.9.3.4 Asociar datos a pozos

Esta ventana que permite asociar indistintamente un conjunto de observaciones o un ensayo de bombeo a un pozo y también mostrar los conjuntos de observaciones y ensayos ya vinculados a éste, cuenta básicamente con tres instancias de **QtGui.QTableView** y cuatro botones que son instancias de **QtGui.QPushButton**.



Cuando se abre la ventana, al igual que sucede con la pantalla mencionada en el apartado anterior, para cargar las observaciones y ensayos se instancian dos objetos de la clase **modelotabla** que luego se utilizarán para setear los modelos de las dos QTableViews correspondientes. Los datos que se carguen en la tercera tabla (QTableView) van a depender de cuáles de los QPushButton superiores se presionen. En caso de presionar el botón Asociar observación, se listarán en esta tabla los conjuntos de observaciones que todavía no han sido asociadas a ningún pozo instanciando un nuevo **modelotabla** para tal fin. En caso de presionar Asociar ensayo de bombeo, se listarán los ensayos de bombeo de la misma forma.

Por tal motivo es que se utilizaron las siguientes variables de estado dentro de esta ventana:

tipo

Es una bandera que puede tomar los valores “e” u “o” y que se utiliza para determinar qué tipo de objetos se asociarán al pozo seleccionado.
Si guarda el valor “o” significa que se vinculará con un conjunto de observaciones. Si se guarda el valor “e” significa que se vinculará con un ensayo de bombeo.

oe

Es una variable que almacena temporalmente el objeto o ítem seleccionado, la cual será seteada luego de que se active la señal de clickeo sobre una celda de la tercera tabla mencionada.

Ambas variables serán útiles en el momento de presionar el botón Agregar, el cual permitirá definitivamente crear las asociaciones necesarias.

4.10 Testing

El proceso de testing permite verificar y evaluar la calidad, factibilidad del producto de software de la misma manera que la viabilidad del proyecto en general y sus funcionalidades.

Este proceso tiene como referencia a una de las áreas de la Ingeniería de Software, en donde provee una variedad de modelos referentes según al tipo de arquitectura, extensibilidad y amabilidad. Dentro de esta área se utilizó como modelo a seguir, la validación y verificación de los requisitos, donde la correcta y completa comunicación con el Ing. Pablo Gamazo determinó la base fundamental para una correcta verificación y validación de los requisitos.

Según la arquitectura de este proyecto que se muestra en una sección siguiente de este documento, donde se especifica un diseño ampliamente modular es necesario determinar un plan de evaluación y testing.

Este plan detalla que cada prueba de verificación y validación entre los distintos módulos del sistema, sean realizadas a medida que se desarrollan. Es aquí donde se encuentran módulos que necesiten comunicarse y obtener una respuesta de otro módulo, el cual puede no haber llegado a implementarse aún. De esta manera se debe simular esa respuesta para poder implementar cuál sería el comportamiento final.

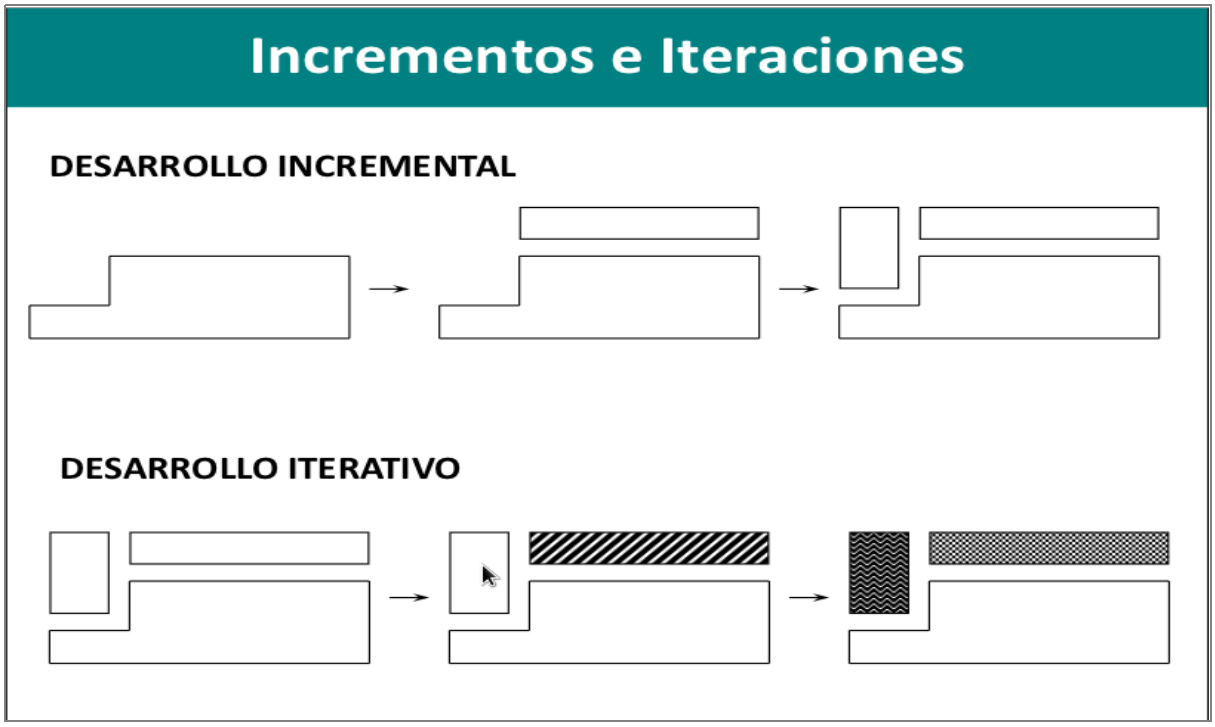
Una vez finalizada esa etapa de pruebas denominadas “pruebas unitarias” se realizan pruebas de integración de los módulos bajo la técnica Bottom-UP [1].

1 Estrategia de integración incremental, la cual comienza por los módulos que no requieren de ningún otro para ejecutarse y se sigue hacia arriba según la jerarquía “usa”

5 Organización del Proceso

5.1 Metodología

El modelo de proceso elegido fue el Iterativo Incremental, el cual hubimos aplicado sistemáticamente en los últimos semestres de la carrera.

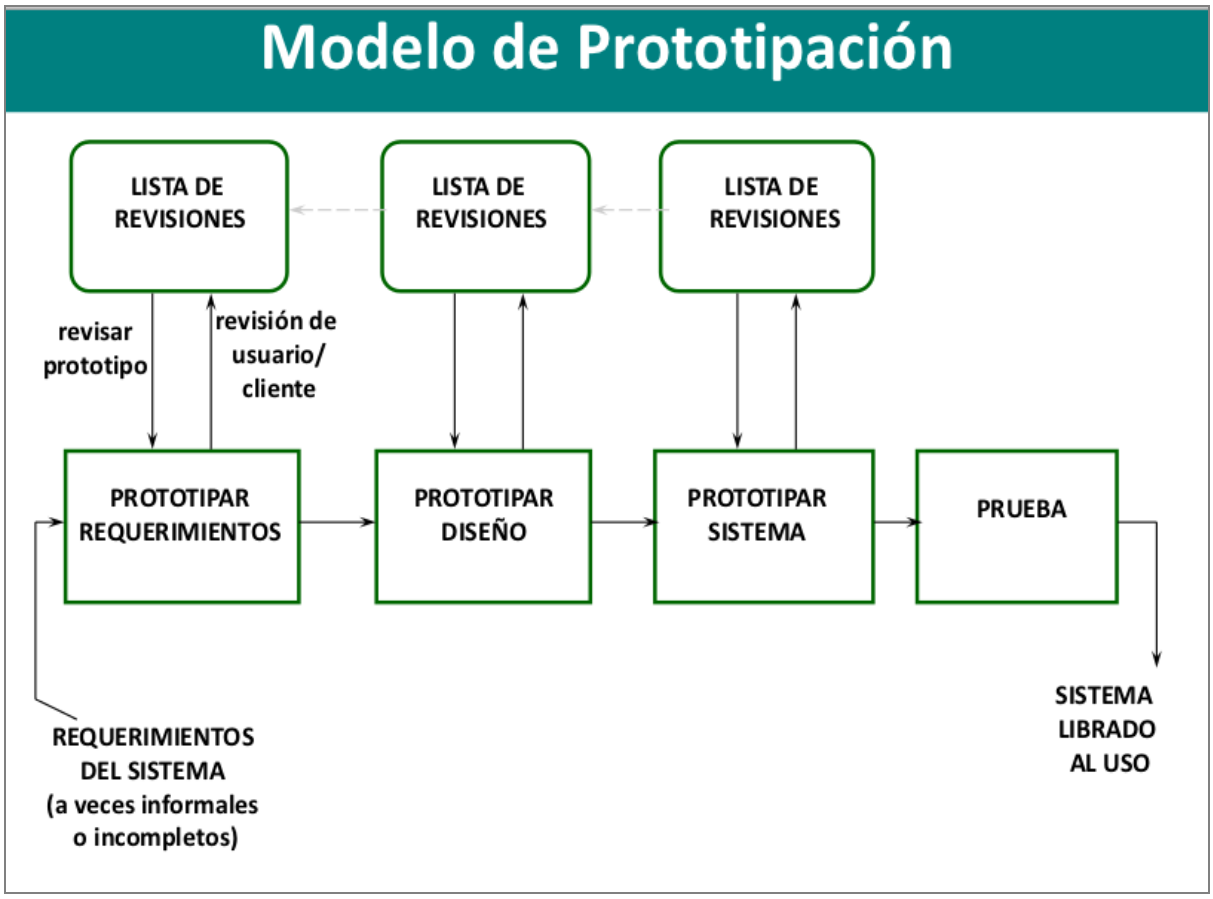


(El desarrollo Iterativo e Incremental, en esta figura se muestra la naturaleza de cada concepto. [1])

Hemos tenido siempre una etapa central; al comienzo la investigación de las herramientas, junto con la aclaración de los conceptos técnicos del área hidrológica. Luego se le sumó la documentación. Nuevos bloques en el análisis, diseño e implementación fueron agregándose posteriormente.

También un modelo de prototipado fue implementado para generar una comunicación con resultados visibles, al momento de iterar con los compañeros de la Regional Norte.

1 Véase la referencia 41 del capítulo “Referencias” del documento del documento “Glosario, apéndice y bibliografía”



(Modelo de prototipación. Se muestran los diferentes prototipos liberados junto con las necesarias revisiones en cada etapa. [1]*)*

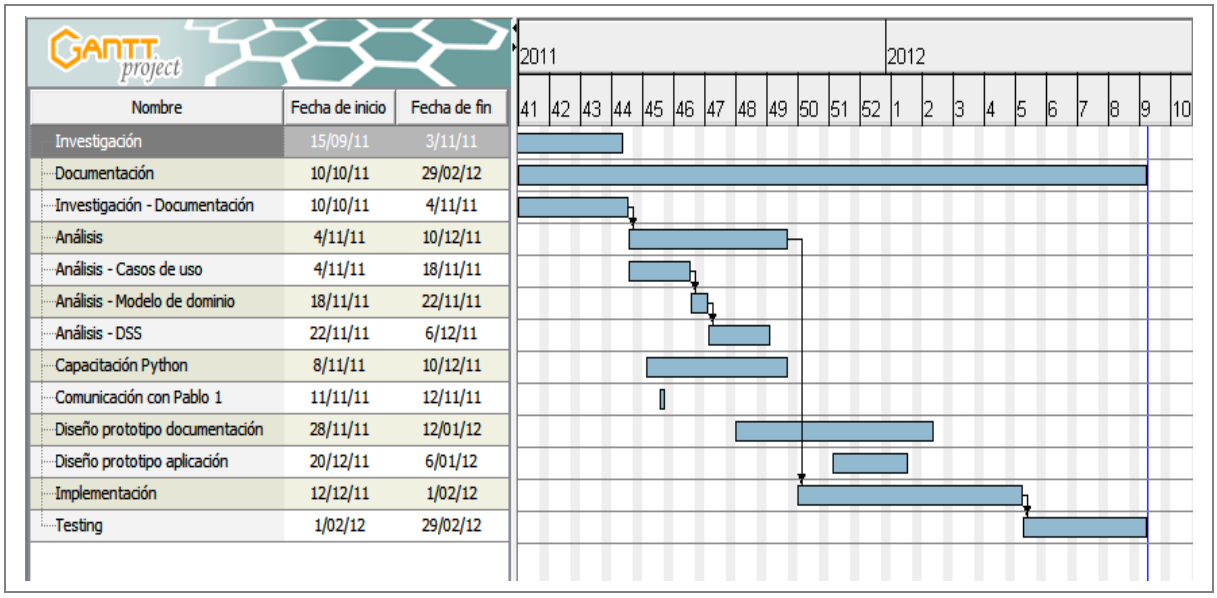
En las diferentes etapas siempre hemos rendido cuentas a nuestros compañeros de Salto, presentando informes tempranos de la investigación, luego documentación con análisis, y diseño, hasta la creación de prototipos del sistema.

Los roles de cada integrante han sido repartidos y llevados a cabo cooperativamente. Cada uno hizo un poco de cada bloque de desarrollo, con el correspondiente líder para cada situación; los líderes más destacados fueron los que tomaron el mando en la investigación hidrológica, y luego en la documentación. En el resto de las actividades el desempeño fue realizado con iguales esfuerzos.

Cabe mencionar, que para la resolución de este problema fue muy importante constar con un equipo como el nuestro, en el que somos cinco y cada uno de sus integrantes aportó el máximo de sus esfuerzos en todas las etapas. Las etapas de investigación se hicieron en dos fases: investigación individual y luego consenso general, lo que aportó mayor riqueza en esta etapa, ayudó a nutrirnos a todos en conocimientos a partir de compartir la información y nos fortaleció en cuestiones de trabajo en equipo. De forma independiente buscamos soluciones a diferentes problemas para luego discutir entre todos la mejor forma de llevar a cabo estos procedimientos.

5.2 Cronograma

A continuación se detalla el cronograma estimado para las diferentes etapas del proyecto y fechas de entrega aproximadas.



Nombre	Fecha de inicio	Fecha de fin
Investigación	15/09/11	3/11/11
Documentación	10/10/11	29/02/12
Investigación - Documentación	10/10/11	4/11/11
Análisis	4/11/11	10/12/11
Análisis - Casos de uso	4/11/11	18/11/11
Análisis - Modelo de dominio	18/11/11	22/11/11
Análisis - DSS	22/11/11	6/12/11
Capacitación Python	8/11/11	10/12/11
Comunicación con Pablo 1	11/11/11	12/11/11
Diseño prototipo documentación	28/11/11	12/01/12
Diseño prototipo aplicación	20/12/11	6/01/12
Implementación	12/12/11	1/02/12
Testing	1/02/12	29/02/12

5.3 Herramientas

5.3.1 Control de versionado

El proyecto en su conjunto fue realizado por cada alumno en computadores del centro de estudio, y en computadores particulares de cada uno; dada esta multiplicidad, es que el control de las versiones y modificaciones a éste se tornó en un punto crítico de consideración.

El grupo ya ha tenido experiencia de trabajo mediante repositorios. Es por esto que no se demoró la decisión de utilizar uno, en particular el servicio de github, lo que nos permitió tener un proyecto central, alojado en el servidor, el cual podía ser descargado desde cualquier computador en el cual se estuviese operando.

5.3.2 Políticas de control de versionado

Haciendo uso de conceptos de mínimos elementos, dentro del vasto abanico de posibilidades que brinda git se estableció la siguiente política de versionado.

1 El proyecto debe de tener una rama central, denominada master. Esta rama o versión es la que se encuentra en el servidor.

2 Cada usuario para trabajar en el proyecto debe de crear una rama distinta a la central y actualizar desde el servidor o modificar lo existente en el computador desde esta rama alternativa.

3 Una vez que el trabajo hecho en esta rama alternativa, satisfizo las pretensiones del usuario, se deben de confirmar dichas modificaciones.

4 Al final del día, o al final de cada momento de trabajo en el proyecto el usuario debería de pasar los cambios confirmados en la rama alternativa, hacia la rama central de trabajo.

5 Las actualizaciones hacia el repositorio en github.com deben de hacerse desde la rama central.

6 Las actualizaciones desde el repositorio en github.com deben de hacerse desde una rama alterna.

Los comentarios adjuntos con cada confirmación deben de estar relacionados a lo hecho.

5.3.3 Herramientas Colaborativas

Las herramientas colaborativas tienen como principal destaque, facilitar el trabajo y organización de las personas a través de la red.

Entre lo más destacado está el hecho de poder crear un documento de tipo Office, y compartirlo con determinados usuarios, cada uno con determinados privilegios, los cuales podrían, en potencia, estar editándolo y viéndolo al mismo tiempo.

Se aprovechó el servicio de Gmail, Google Docs, para crear documentos compartidos útiles en cada etapa del proyecto, desde documentos propios de la Documentación, hasta croquis, actas de las conversaciones con Pablo Gamazo o apuntes referentes a cada tema de discusión importante para el grupo.

5.3.4 Herramientas de desarrollo

5.3.4.1 IDE'S

El hecho de que Python puede ser desarrollado sin la necesidad de grandes esfuerzos de configuración, se decidió no establecer como necesario ningún IDE para el desarrollo del proyecto.

Herramientas como gEdit [1] o NotePad++ [2], disponibles en Windows o Linux fueron suficientes para codificar. Ambas presentan un sistema de identificación de elementos dentro del código mediante colores para una variedad de lenguajes, entre ellos Python. También estos editores pueden mostrar visiblemente las tabulaciones y los espacios en blanco que se dejan en los archivos, lo cual ha sido de gran utilidad, dado a que el intérprete de Python toma decisiones dependiendo de la organización de las sentencias, asignaciones y órdenes de ejecución.

1 Véase la referencia 38 del capítulo “Referencias” del documento del documento “Glosario, apendice y bibliografía”
2 Véase la referencia 37 del capítulo “Referencias” del documento del documento “Glosario, apendice y bibliografía”

5.4 Dificultades del proceso

Como parte natural de un proceso de desarrollo, el individuo o el grupo, se ha de encontrar con obstáculos a superar en cualquier momento del transcurso. Nuestra situación no fue la excepción; ante los nuevos retos y el poco tiempo de duración que del mismo teníamos, por supuesto que surgieron nuevas dificultades en los diferentes ámbitos. Las más destacadas fueron las descritas a continuación.

Desde el primer momento, teníamos presente que íbamos a tener que mantener una comunicación permanente con el grupo de Hidrología Subterránea de la Regional Norte, quienes son las personas que están más interiorizadas en las temáticas para las cuales esta aplicación dará soporte. En la agenda y planificación de reuniones con Pablo Gamazo, surgió la común problemática de encontrar días y horarios para generar el intercambio. Dada la distancia física entre Paysandú y Salto, el contacto se mantuvo mediante comunicación virtualya fuese por mails o charlas por Skype, este proceso comunicativo se extendió a lo largo del proyecto.

Por el contenido técnico hidrológico del proyecto, antes de conformar una idea clara sobre la realidad del problema, se debió invertir tiempo en la investigación de cual era el dominio sobre el que debíamos trabajar. Esto incluye la aclaración de detalles y terminologías técnicas exclusivas de la hidrología. Para la comprensión del problema y la búsqueda de soluciones fueron muy importantes los constantes intercambios con el grupo mencionado, de los cuales estuvimos pendientes por nuestra falta de conocimientos en profundidad en esta materia.

El detalle técnico también transigió al proyecto un contenido matemático específico de la materia, el cual fue absorbido por el grupo de manera satisfactoria. No obstante, al ser este un grupo con aptitudes mas orientadas hacia el desarrollo de soluciones informáticas, el entendimiento de la teoría matemática también requirió buena parte de nuestro tiempo, para comprender su naturaleza y funcionamiento.

En la elección de lenguajes y herramientas, se llegó a la conclusión que Python brindaba el ámbito más óptimo para elaborar el proyecto. En este sentido, el problema fue que este lenguaje era desconocido para el grupo, lo que conllevó a aprender su sintaxis, forma de codificación y ejecución. También se necesitó inquirir sobre la interfaz gráfica elegida, PyQt4, y el funcionamiento de la herramienta de graficado Matplotlib.

6 Conclusión

Para comenzar este capítulo es importante señalar la complejidad del problema abordado, teniendo en cuenta el perfil de nuestra formación, ante lo cual se considera que el software desarrollado conformó al cliente final y a los propios integrantes del grupo.

De acuerdo a los requerimientos obtenidos de la Propuesta de Proyecto, y a la primera reunión con el cliente, se comenzó a investigar sobre las diferentes herramientas y lenguajes del ámbito informático que fueran adecuadas para brindar soluciones al problema. En esta temprana etapa se comenzó con la tarea de documentación, realizando posteriormente la introducción a los conceptos más generales sobre los acuíferos, los cuales fueron parte del ámbito de nuestro proyecto.

A partir de lo ya expresado, mediante una comparativa de herramientas, se estableció que la cobertura de las exigencias de los requerimientos por parte de Python, hicieron acertada su elección como lenguaje para el desarrollo, mientras que el intercambio de información y datos con el cliente permitió la evolución de los requerimientos[1].

Al momento del modelado del dominio, se consideró la creación de un objeto dominio que sería el encargado de contener las relaciones con los diferentes elementos involucrados en cada nuevo proyecto y se decidió contar con un objeto solucion del cual heredasen los diferentes métodos de solución, ya sea analíticos o numéricos. Otra consideración importante fue la separación de los parámetros de las soluciones. El objetivo en este caso fue diseñar un modelo que facilitase la extensibilidad del sistema, brindando de esta manera un procedimiento bien definido para que, en etapas posteriores, se agreguen nuevos métodos de solución, suministrando una forma común de acceso y adición de los parámetros del dominio[2].

En lo concerniente a la implementación de cálculos, hubo que comprender primero los scripts[3] enviados por el grupo de la Regional Norte, codificados en Matlab[4], para luego encontrar su correcta traducción al lenguaje utilizado. Además de esto, se debieron implementar dos conceptos importantes en el sistema al momento de la interpretación de los ensayos de bombeo. En primer lugar, se determinó que el agua en un acuífero se mantendrá en un determinado nivel fijo conocido[5] antes de que se efectúe el correspondiente ensayo. En el cálculo de los diferentes métodos de solución se consideran como datos conocidos estos niveles de agua iniciales en todos los puntos del dominio, por lo que se implementó un cálculo involucrando los coeficientes a, b y c, en un principio ingresados al momento de crear un nuevo proyecto.

1 Para una mejor comprensión de la evolución de requerimientos en nuestro proyecto, ver punto 2.2 Evolución de requerimientos, del capítulo 2 Definición del problema.
2 Para una mejor comprensión del modelo, ver punto 3.2 Modelo dominio, del capítulo 3 Solución al problema.
3 Véase la definición de script en el capítulo “Glosario” del documento “Glosario, apéndice y bibliografía”.
4 Véase capítulo “Herramientas científicas existentes”, en el documento “Estado del arte”.
5 Para una explicación detallada véase punto 4.9.2 Implementación de cálculos, Niveles iniciales.

En segundo lugar, se consideró la importancia y el fin de utilizar postprocesos[6] en la aplicación, para garantizar la correcta interpretación de los ensayos de bombeo en nuestro sistema. Se observaron los postprocesos en los scripts en Matlab y se buscó dar una solución similar en lo conceptual, pero orientada a objetos y adaptada a nuestro sistema. La concreción de estos puntos garantizó la correcta interpretación de los ensayos en nuestro sistema.

Lo anteriormente mencionado fue fundamental para que el proyecto fuese extensible, siendo uno de los requerimientos previos, y para que el graficado de los ensayos presentara los mismos resultados que los vistos en Matlab.

Se concluye que los requerimientos funcionales y no funcionales fueron cumplidos, lográndose la manipulación de elementos, la asociación de datos y el procesamiento de los cálculos, dejando a su vez una aplicación extensible, con el código fuente debidamente comentado, para las agregaciones futuras.

6 Para una explicación detallada véase punto 4.9.2 Implementación de cálculos, Postprocesos.