Instituto Tecnológico de San Juan del Río

S.P.R.E.S.A.C

Sistema para el registro de entradas y salidas de areas criticas

Autor:
J. Carlos Ávila

Supervisor:
MGTI Rogelio Vázquez

25 de enero de 2016

Índice

"El auténtico genio consiste en la capacidad para evaluar información incierta, aleatoria y contradictoria."
Winston Churchill, estadista.

Introducción

Construir una base de datos es una tarea ardua, llena de trabajo manual y repetitivo en la mayoría de las veces, al menos así lo ha sido por los últimos miles de años, una costumbre o mas bien, una necesidad que se empezó a estandarizar y dejar en manos de expertos, ya lo hacían los Babilonios en sus tablillas de arcilla encontradas en las ruinas de en medio oriente, después de ellos los Egipcios y así sucesivamente a lo largo de la historia de la humanidad.

Mantener un control, registro y más que nada mantenerse informado de como de comporta cierto fenómeno, mismo que puede ser cualquier cosa o evento que presente cierto interés para un grupo en especifico.

Con el tiempo las técnicas fueron avanzando paulatinamente, desde las antiguas tablillas de arcilla, pasando por los papiros, grabado en piel, metales y posteriormente el uso del papel como el medio mas accesible para el almacenamiento de información, fuera de la clase que fuese. Cada vez los métodos se perfeccionaron y entonces en algún punto del siglo XX se invento una de las maquinas más versátiles de la historia humana, si bien en un principio su principal objetivo era el de ayudar en la resolución de complejos cálculos matemáticos, pronto ser descubrieron muchas mas capacidades de estos dispositivos, uno de esos fue el procesamiento de datos que arrojaran resultados concretos en poco tiempo, el tratamiento de grandes cantidades de información se facilito en gran medida, después de eso la industria cambio por completo y la cantidad de datos que se manejaban se multiplico en relación directamente exponencial, las capacidades de las cada vez más eficientes y poderosas COMPUTADORAS no han detenido su avance hasta nuestros días.

En nuestro tiempo actual es prácticamente inconcebible el manipular la datos de forma manual, como se hacia antiguamente, y es que no es que no se sepa como hacerlo es que simple se hace imposible en términos de eficiencia, pues la cantidad de datos que hay que procesar el increíblemente grande, en absoluto se compara, es por ello que las computadoras son las que se encargan de ello en la actualidad, claro, ello no quiere decir que las interferencia humana en el proceso se haya eliminado, pues al final siempre debe haber alguien que controle y le diga como hacerlo, ahí es donde entra en juego el papel de los desarrolladores de software.

Es por ello que el siguiente proyecto se centra en actualizar la forma en que se recolecta, almacena y procesa la información de áreas en las que a estas fechas dicha

actividad se hace de manera manual y de forma tradicional, con las limitaciones que presenta.

1. Objetivos

Desarrollar un sistema que permita el registro de entradas y salidas de las diferentes áreas que así lo requieran, mediante la construcción de un núcleo operativo sólido y robusto que almacene y gestione la información con el fin de llevar un control detallado de los movimientos en las áreas en las que se implante.

1.1. Objetivos específicos

- Construir módulos diseñados para la recolección de información deseada.
- Manipular la información para obtener información relevante, en los siguientes puntos:
- Tener un registro del movimiento de personal en áreas específicas.
- Mantener un registro de la entrada de alumnos al plantel, así como de externos y profesores.
- Implementar un control de asistencia a conferencias y talleres que se lleven a cabo en las instalaciones del ITSJR.
- Enlazar el sistema con la base de datos del SII, para tener la información de alumnos y profesores actualizada.
- Reforzar la seguridad general de la institución, con base en los registro de movimientos.
- Agilizar los procesos de registro de acceso.

2. Marco teórico

2.1. Metodología de desarrollo extremo

La programación extrema se basa en una serie de reglas y principios que se han ido gestando a lo largo de toda la historia de la ingeniería del software. Usadas conjuntamente proporcionan una nueva metodología de desarrollo software que se puede englobar dentro de las metodologías ligeras, que son aquéllas en la que se da prioridad a las tareas que dan resultados directos y que reducen la burocracia que hay alrededor tanto como sea posible (pero no más) . La programación extrema, dentro de las metodologías ágiles, se puede clasificar dentro de las evolutivas.

Una de las características de *eXtreme Programming* es que muchos de, si no todos, sus ingredientes son de sobra conocidos dentro de la rama de la ingeniería del software desde hace tiempo, incluso desde sus comienzos. Los autores de han seleccionado los que han considerados como los mejores y han profundizado en sus relaciones y en cómo se refuerzan unos a otros. El resultado ha sido una metodología única y compacta. Por eso, aúnque se pueda alegar que la programación extrema no se base en principios nada nuevos, se ha de aclarar que, en conjunto, es una nueva forma de ver el desarrollo de software.

2.1.1. El proceso de desarrollo extremo

La programación extrema parte del caso habitual de una compañía que desarrolla software, generalmente software a medida, en la que hay diferentes roles: un equipo de gestión, un equipo de desarrolladores y los clientes. La relación con el cliente es totalmente diferente a lo que se ha venido haciendo en las metodologías tradicionales que se basan fundamentalmente en una fase de captura de requisitos previa al desarrollo y una fase de validación posterior al mismo.

2.1.2. Interacción con el cliente

En la programación extrema al cliente no sólo se le pide que apoye al equipo de desarrollo, en realidad podríamos decir que es parte de él. Su importancia es capital a la hora de abordar las historias de los usuarios y las reuniones de planificación, como veremos más adelante. Además, será tarea suya realimentar al equipo de desarrolladores después

de cada iteración con los problemas con los que se ha encontrado, mostrando sus prioridades, expresando sus sensaciones... Existirán métodos como pruebas de aceptación que ayudarán a que la labor del cliente sea lo más fructífera posible.

En resumen, el cliente se encuentra mucho más cercano al proceso de desarrollo. Se elimina la fase inicial de captura de requisitos y se permite que éstos se vayan definiendo de una forma ordenada durante el tiempo que dura el proyecto. El cliente puede cambiar de opinión sobre la marcha y a cambio debe encontrarse siempre disponible para resolver dudas del equipo de desarrollo y para detallar los requisitos especificados cuando sea necesario.

El proceso de captura de requisitos de XP gira entorno a una lista de características que el cliente desea que existan en el sistema final. Cada una de estas características recibe el nombre de historias de usuarios y su definición consta de dos fases:

En la primera fase el cliente describe con sus propias palabras las características y el responsable del equipo de desarrollo le informa de la dificultad técnica de cada una de ellas y por lo tanto de su coste.

La segunda fase consiste en coger las primeras historias que serán implementadas (primera iteración) y dividirlas en las tareas necesarias para llevarlas a cabo.

Este proceso es una de las principales diferencias con las metodologías tradicionales. aúnque las historias de usuarios guardan cierta relación con otras técnicas como los casos de uso de UML, su proceso de creación es muy diferente. En lo que al cliente se refiere no se le exige que especifique exactamente lo que quiere al principio con un documento de requisitos de usuario. La parte que se mantiene con este documento es que es el cliente el que tiene que escribir lo que quiere, no se permite que alguien del equipo de desarrolladores lo escriba por él.

2.1.3. Planificación del proyecto

La planificación debe de seguir unas ciertas premisas. La primordial es que las entregas se hagan cuanto antes y que con cada iteración el cliente reciba una nueva versión. Cuanto más tiempo se tarde en introducir una parte esencial, menos tiempo habrá para trabajar en ella posteriormente. Se aconsejan muchas entregas y muy frecuentes. De esta forma, un error en una parte esencial del sistema se encontrará pronto y, por tanto, se

podrá arreglar antes.

Sin embargo, los requisitos anteriores en cuanto a la planificación no deben suponer horas extra para el equipo de desarrollo.

Pero lo mejor de todo es que a la hora de planificar uno se puede equivocar. Es más, todos sabemos que lo común es equivocarse y por ello la metodología ya tiene previsto mecanismos de revisión. Por tanto, es normal que cada 3 a 5 iteraciones se tengan que revisar las historias de los usuarios y renegociar nuevamente la planificación.

2.1.4. Diseño, desarrollo y pruebas

El desarrollo es la pieza clave de todo el proceso de programación extrema. Todas las tareas tienen como objetivo que se desarrollo a la máxima velocidad, sin interrupciones y siempre en la dirección correcta.

También se otorga una gran importancia al diseño y establece que éste debe ser revisado y mejorado de forma continua según se van añadiendo funcionalidades al sistema

La clave del proceso de desarrollo de XP es la comunicación. La gran mayoría de los problemas en los proyectos de desarrollo son provocados por falta de comunicación en el equipo, así que se pone un gran énfasis en facilitar que la información fluya lo más eficientemente posible.

Como ya hemos visto con anterioridad, uno de los principios de la programación extrema es la simplicidad. El diseño debe ser lo más simple posible, pero no más simple. El paradigma KISS ("Keep It Small and Simple"para unos o "Keep it Simple, Stupid"para otros) se lleva hasta las últimas consecuencias. Por ejemplo, se hace énfasis en no añadir funcionalidad nunca antes de lo necesario, por las sencillas razones de que probablemente ahora mismo no sea lo más prioritario o porque quizás nunca llegue a ser necesaria.

Supongamos que ya hemos planificado y dividido en tareas, como se ha comentado en los párrafos anteriores. Lo lógico sería empezar ya a codificar. Pues no. Nos encontramos con otro de los puntos clave de la programación extrema (y que sí es innovador en ella): las pruebas unitarias se implementan a la vez hay que el código de producción. De hecho cada vez que se va a implementar una pequeña parte se escribe una prueba

sencilla y luego el código suficiente para que la pase. Cuando la haya pasado se repite el proceso con la siguiente parte.

Esta forma de usar las pruebas unitarias ayuda a priorizar y comprobar la evolución del desarrollo y que ofrecen realimentación inmediata. Ya no hay imprescindibles dos equipos diferenciados que desarrollan y prueban cada uno por su cuenta. Ahora el ciclo se basa en implementar una prueba unitaria, codificar la solución y pasar la prueba, con lo que se consigue un código simple y funcional de manera bastante rápida. Por eso es importante que las pruebas se pasen siempre al 100

Las pruebas unitarias no se han de confundir con las pruebas de aceptación que han sido mencionadas con anterioridad. Éstas últimas son pruebas realizadas por el cliente o por el usuario final para constatar que el sistema hace realmente lo que él quiere.

La programación extrema viene a perseguir lo que se ha venido a llamar integración continua. De esta forma, haciéndolo cada vez con pequeños fragmentos de código, se evita la gran integración final.

En todo desarrollo de programación extrema debería existir, por tanto, una versión siempre integradaLa sincronización por parte de los desarrolladores con el repositorio central debe darse como mínimo una vez al día, de manera que los cambios siempre se realicen sobre la última versión. De esta forma nos podemos asegurar de que las modificaciones que hacemos no se estén haciendo sobre una versión obsoleta

El proceso de desarrollo no lo va a hacer un desarrollador en solitario, sino siempre con otra persona, algo que se ha venido a llamar programación por parejas. Una pareja de desarrolladores debe compartir ordenador, teclado y ratón. El principal objetivo es realizar de forma continua y sin parar el desarrollo una revisión de diseño y de código. Las parejas deben ir rotando de forma periódica para hacer que el conocimiento del sistema se vaya difundiendo por el equipo (facilitándose que el código sea de todos), a la vez que se fomentan el entrenamiento cruzado.

2.1.5. Resumen de XP

Todos los punto anteriormente detallados, bien los podemos definir y resumir en la siguiente lista, obviamenta cada uno de ellos lleva consigo un gran significado de fondo, pero en general y conociendo la metodología se puede uno basar en la siguiente lista para cerciorarse de que se estan cumpliendo los puntos que la metodología estipula que

se deben llevar a cabo.

1. El juego de la planificación (the planning game).
2. Pequeñas entregas (small releases).
3. Metáfora (metaphor).
4. Diseño simple (simple design).
5. Pruebas (testing).
6. Refactorización (refactoring).
7. Programación por parejas (pair programming).
8. Propiedad colectiva (collective ownership).
9. Integración continua (continous integration).
10. 40 horas semanales (40-hour week).
11. Cliente en casa (on-site costumer).
12. Estándares de codificación (coding standards).

Se puede considerar como el resumen de bolsillo de la metodología de desarrollo extremo, y llevarla a cada lugar en el que se este desarrollando.

2.2. Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License,1 que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

2.2.1. Características y paradigmas

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones.

Python usa tipado dinámico y conteo de referencias para la administración de memoria.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

Aunque la programación en Python podría considerarse en algunas situaciones hostil a la programación funcional tradicional del Lisp, existen bastantes analogías entre Python y los lenguajes minimalistas de la familia Lisp como puede ser Scheme.

2.2.2. Filosofía

Los usuarios de Python se refieren a menudo a la Filosofía Python que es bastante análoga a la filosofía de Unix. El código que sigue los principios de Python de legibilidad y transparencia se dice que es "pythonico". Contrariamente, el código opaco u ofuscado es bautizado como "no pythonico"(ünpythonic.[en] inglés). Estos principios fueron famosamente descritos por el desarrollador de Python Tim Peters en El Zen de Python

2.3. PyQt

PyQt es un conjunto de herramientas para crear aplicaciones Graficas (GUI). Es una mezcla entre lenguaje de programación Python y libreria Qt. La libreria Qt es una de las liberias graficas mas poderosas del planeta. En lo personal creo que es facilmente mas poderosas que Tkinter. El sitio oficial para PyQt. Fue desarrollado por Phil Thompson.

PyQt se implemento como un conjunto de módulos para python. Cuenta con más de 300 clases y casi 6000 funciones y métodos. Se trata de un conjunto de herramientas

multiplataforma. Funciona en todos los sistemas operativos más importantes. Incluyendo UNIX, Windows y Mac. PyQt tiene licencia doble. Los desarrolladores pueden elegir entre la licencia GPL y una licencia comercial. Anteriormente, la licencia GPL sólo está disponible en Unix. Pero a partir de la versión PyQt 4, la licencia GPL está disponible en todas las plataformas soportadas.

Debido a que hay una gran cantidad de clases disponibles, se han dividido en varios módulos, como ilustra la imagen 1.

Figura 1: Modulos de PyQt

3. Desarrollo

3.1. Análisis de requerimientos de usuario

Automatizar la tarea de registro de entrada y registro de salida del laboratorio de computo del Instituto Tecnológico de San Juan del Río, es algo que hasta día de hoy se hace de manera manual, anotando en formatos físicos *papel*, la hora de entrada del mismo, en general son los prestadores de servicio social los que se encargan de realizar dicha actividad, que ademas de ser repetitiva y con doble carga de trabajo, pues una vez realizado el registro este, a un tiempo determinado tiene que ser contabilizado y transferido a una base de datos de forma manual.

Como se puede apreciar es un proceso repetitivo y bien definido, que no obstante es propenso a errores humanos o en su defecto a la corrupción del medio físico en el cual se realiza el registro de la información, que ademas es muy limitada en tanto a los datos que provee.

Por esa y algunas otras razones que posiblemente se nos escapen ahora, es que la construcción de un sistema que automatice el proceso y que ademas agregue otras funciones deseables, se ve no solo como algo que debe ser implementado sino como una evolución natural del sistema anterior.

Para lograr implementar de forma exitosa el proyecto es necesario que el se cubran los requerimientos funcionales y no funcionales que a continuación se detallan, una vez que todos y cada uno de ellos se haya cumplido e integrado en un sistema funcional que cumpla con las especificaciones propuestas, se podrá decir que el sistema esta listo para su uso.

3.2. Análisis Requerimientos técnicos funcionales

Recolectar información es uno de los objetivos primarios, para ello es necesario que el sistema permita realizar ciertas tareas especificas y bien definidas, y por medio de ellas obtener el resultado esperado. Cada una de la cuales representara un modulo individual, y como la metodología de desarrollo que se utiliza 2.1 se fundamenta en el desarrollo de prototipos, y iteraciones por cada uno de estos, sera esta pues, la forma en que se trabajara.

Comenzaremos describiendo cada uno de los módulos y su función, iniciaremos por los que están mas cercanos al usuario final, que en este caso son los alumnos y el perso-

nal que administre el sistema.

3.2.1. Interfaz de usuario

Una interfaz gráfica de usuario **GUI** *(por sus siglas en ingles)* que permita la interacción directa con el sistema, clara y fácil de usar. Su principal objetivo es que permita al usuario ingresar los datos necesarios para que los diferentes módulos trabajen, del lado de los alumnos la única información requerida es su numero de control y el numero de maquina que ha de usar, ordenar que se genere el registro y listo. El resto del trabajo se hace debajo dela capa de abstracción que representa dicha interfaz.

En la misma aparte de los campos para numero de control y maquina, también se deberá desplegar un mapa o diagrama de la distribución de los equipos, en donde de pueda ver los equipos disponibles y los que estén en ocupados en determinado momento.

Todo esto sin olvidar la estética, usabilidad y confiabilidad con la que debe de contar para superar las pruebas de calidad.

3.2.2. Interfaces de adminsitración

Ya que se han construido la interfaz para el usuario, se deberá crear un panel de administración, misma que permita configurar ciertos parámetros, realizar consultas y motitorear el funcionamiento del sistema en general.

En esta primera version funcional dicha interfaz no contara con todos los controles disponibles, gradualmente conforme se avance en el desarrollo del proyecto se irán agregando funcionalidades cada que estén listas y se acoplen perfectamente con el resto del sistema, de acuerdo al funcionamiento de la metodología que se ha implementado en el desarrollo.

3.2.3. Validar

Las interfaces de usuario y de administración alimentan al sistema para que funcione, sin embargo la información que se introduce debe de ser valida, y cumplir con ciertos requisitos, de lo contrario al intentar procesar los datos estos que el sistema se salga de los establecidos y por lo tanto cause que el mismo falle y en el peor de los casos termine su ejecución con la posible perdida de información.

Para evitar este tipo de fallo es necesario implementar mecanismos que permitan validar que la información que se introduce es la correcta, básicamente son solo dos los

valores que se introducen por parte del usuario, su numero de control y el número de máquina que usara, por lo que los podemos integrar la validación de ambos en un solo modulo que se encargue de analizar y retornar una respuesta al programa principal.

Validar el número de control del alumno consta de los siguientes puntos, mismos que se deben cumplir para que la misma sea exitosa:

- Sintaxis
- Existencia [1]

Como se puede ver, son solo dos puntos, mismo que se eben cumplir sin excepción alguna, pues de lo contrario no puede continuar el proceso de registro.

3.2.4. Obtener datos relacionados

Cuando se valida en numero de control, y ser determina que si sintaxis cumple con las ciertas características, lo siguiente es extraer los datos que estén asociados a ese numero de control, mismo que internamente se maneja como la clave principal de cada uno de los alumnos, a si mismo todos y cada uno de los datos almacenados esta intrínsecamente ligado a una clave. Los datos que se recolectan son los siguientes:

- Número de control
- Nombre
- Apellido Paterno
- Apellido Materno
- Carrera (*Especialidad*)

Por lo que cuando el alumno ingrese su numero de control, el sistema después de validarlo, realiza una consulta a la base de datos, esto con la finalidad en un principio, de determinar que en efecto existe en la misma. Posteriormente si existe, retorna los datos del alumno, mismo que se mostraran en campos especialmente reservados para tal fin el formulario de registro 3.2.1.

Dichas tareas se realizaran de forma transparente al usuario, que apenas detectara que se tarda en efectuarlas.

[1] *Se refiere al echo de si el alumno esta activo o dado de baja en el sistema SII.*

3.2.5. Numero de máquina

Ya se vio que se espera de la validación de los datos que se introducen por parte del usuario en la 3.2.3 se describe. Una vez que se verifica que el alumno exista en la base de datos, se visualizan los datos del mismo en el formulario de registro, de procede a validar la máquina que quieres usar, en este punto se debe comprobar lo siguiente:

- Sintaxis
- Disponibilidad [2]
- Existencia [3]

Si los puntos anteriores se satisfacen con éxito, el proceso continua.

3.2.6. Preparar registro

Una vez que todos los datos han sido validados y recolectados se procede a preparar la consulta, o en otros términos recolectar los datos necesarios para efectuar el registro de entrada o de salida en la base de datos, que si todo el proceso ha sido exitoso no conlleva ninguna complejidad. Los datos en cuestión son los siguientes:

ID : un identificador auto-generado y con incremento automático, para identificar el registro.

Número de control : puede ser el que introduce el usuario o el que se lee de la base de datos.

Número de máquina : la maquina que usara.

Fecha de entrada : en formato `AAAA-MM-DD`.

Hora de entrada : en formato `H:M:S`.

La fecha y hora se obtienen directamente de los que maneja el sistema, una función especial se encarga de obtener dichos datos.

[2] *Si esta libre u ocupada, según corresponda*

[3] *No confundir con el estatus de alumnos, en esta caso indica si esta funcional o no la maquina.*

3.2.7. Restablecer el formulario

Una vez que se ha realizado con éxito el registro de entrada ó salida, se procede a limpiar el contenido de los campos del formulario, y eliminar el contenido de las variables usadas internamente, dejando listo el sistema para un nuevo registro. El sistema entonces entra en estado de espera, hasta que detecta que el campo correspondiente al numero de control ha cambiado, pasando de estar vació a contener alguna cadena de caracteres, entonces comienza el ciclo nuevamente.

3.2.8. Ampliaciones a futuro

Los módulos contenidos hasta este punto, así como su función son los mínimos y estrictamente requeridos para que el sistema funcione de manera básica, posteriormente en futuras iteraciones la funcionalidad del mismo se extenderá considerablemente, obviamente por cada una de ellas que sea integrada se agregara su respectiva sección en este documento.

3.3. Análisis de requerimientos técnicos no funcionales

En esta sección se describen los elementos que si bien no son componentes clave del sistema, son necesarios para que el mismo funcione de forma adecuada, en el estado actual de desarrollo esto son más bien pocos.

3.3.1. Disponibilidad

El periodo de operación debe ser de trece horas cada día hábil de la semana, lo que generalmente es de lunes a viernes de 07:00 a 20:00, lapso en el cual debe estar en pleno funcionamiento y disponibilidad.

Dado que el sistema se instalara en una maquina dedicada únicamente para este fin, dicho equipo deberá ser configurado para solo funcionar para determinas tareas que serán descritas en puntos posteriores.

3.3.2. Accesibilidad

Las interfaces de usuario se han pensado para ser fáciles de usar, y con apenas interacción con el usuario, fuera de las tres pasos que debe realizar para proveer los datos

necesarios al sistema:

1. Ingresar número de control.
2. Seleccionar una máquina.
3. Pulsar la tecla `Enter` para efectuar el registro.

Como se puede ver el nivel de interacción por parte del usuario es muy simple, el diseño de la interfaz es también simple y visualmente confortables. En la figura 2 se puede ver el aspecto de la interfaz de usuario. En donde se puede comprobara que es accesible.

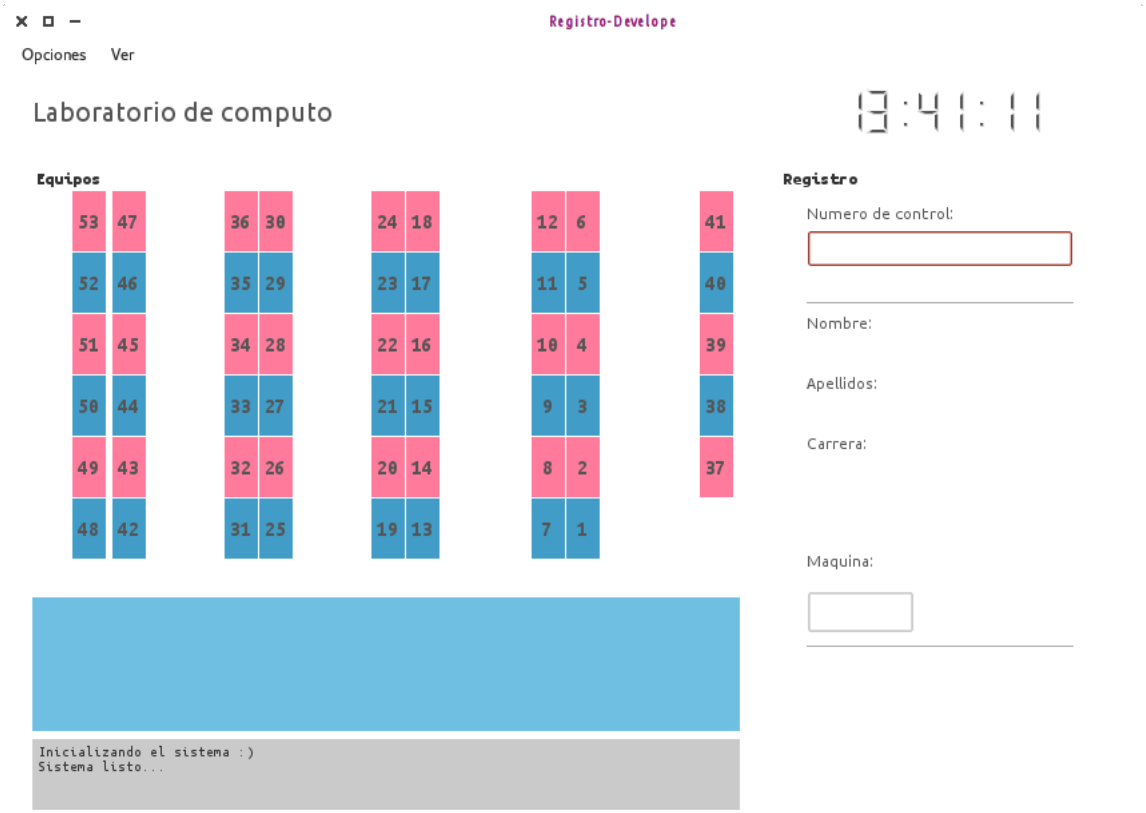

Figura 2: Interfaz de usuario.

3.3.3. Rendimiento

La elección de las plataformas de desarrollo fueron especialmente pensadas para este proyecto, por un lado el lenguaje de programación `Python` 2.2 es ideal para sistemas

modulares que requieran de un desarrollo acelerado, pese a ser un lenguaje interpretado es lo bastante rápido para ser usado en proyectos importantes, no por nada es usado por grandes empresas como Google. Por el lado de las librerías gráficas, QT, aunque mas específicamente PyQt 2.3, que es el port la primera para ser usada con Python, es multi-plataforma y una de las librerías gráficas mas poderosas.

Por ultimo todo esto se instalara en sistema operativo de tipo UNIX, mas específicamente una distribución linux, Manjaro, misma que se selecciono por algunas razones, entre ellas esta la fácil configuración del sistema, la facilidad de uso para los usuarios que no estan acostumbrados a entornos GNU/Linux.

En resumen se puede estar seguro que con la elección de tecnologías mencionadas hasta este punto, el sistema rendirá como se espera y cumplirá sin problema alguno con los requisitos de funcionales previamente documentados 3.2

3.3.4. Escalabilidad, mantenibilidad y portabilidad

Las herramientas y sistema operativo seleccionado, fueron pensadas para proporcionar desde un principio todas estas posibilidades, desde el sideño mismo del sistema, hasta las posibles arquitecturas en las cuales se implemente.

Por el momento solo hay una restricción, no se puede ejecutar en dispositivos móviles.

3.4. Diseño

Una vez identificados los requerimientos funcionales y no funcionales del sistema, crear un modelo de como debería ser cada uno de los componentes del sistema es posible pero más que posible es algo que se debe hacer a vistas de continuar con la siguiente etapa del proceso, desarrollo, implementación y pruebas.

3.4.1. Diseño del núcleo de la aplicación

El núcleo del sistema es relativamente simple, consta de los siguientes pasos:

1. Introducir numero de control.
2. Validar numero de control.
3. Obtener datos relacionados a dicho número de control.

4. Seleccionar equipo.

5. Recolectar y preparar la consulta de registro.

6. Realizar registro.

7. Inhabilitar el equipo seleccionado.

8. Restablecer los formularios.

Esto para el proceso de registro de entrada, aunque el proceso es exactamente el mismo para registrar la salida del mismo, salvo por que en vez de inhabilitar se libera el equipo en cuestión, que vuelve a estar disponible para futuros usuarios.

El diagrama de flujo de la figura 3 ilustra el proceso de registro de un usuario.

3.4.2. Diseño de la interfaz de usuario

La interfaz de usuario es la misma que se muestra en la figura 2, ahora solo describiremos sus elementos:

1. Número de control para introducir el número de control.

2. Datos de alumno cuando se valida que el número de control es valido, se llenan automáticamente con los datos de este.

3. Número de máquina el equipo que se usara.

4. Diagrama del laboratorio muestra la distribución de los equipos en el laboratorio de computo.

5. Estadísticas diarios muestran cierta información sobre el uso del laboratorio a lo largo del día.

6. Log del sistema muestra información relacionada a la operación que se realiza en el momento.

La imagen de la figura muestra los puntos arriba mencionados.

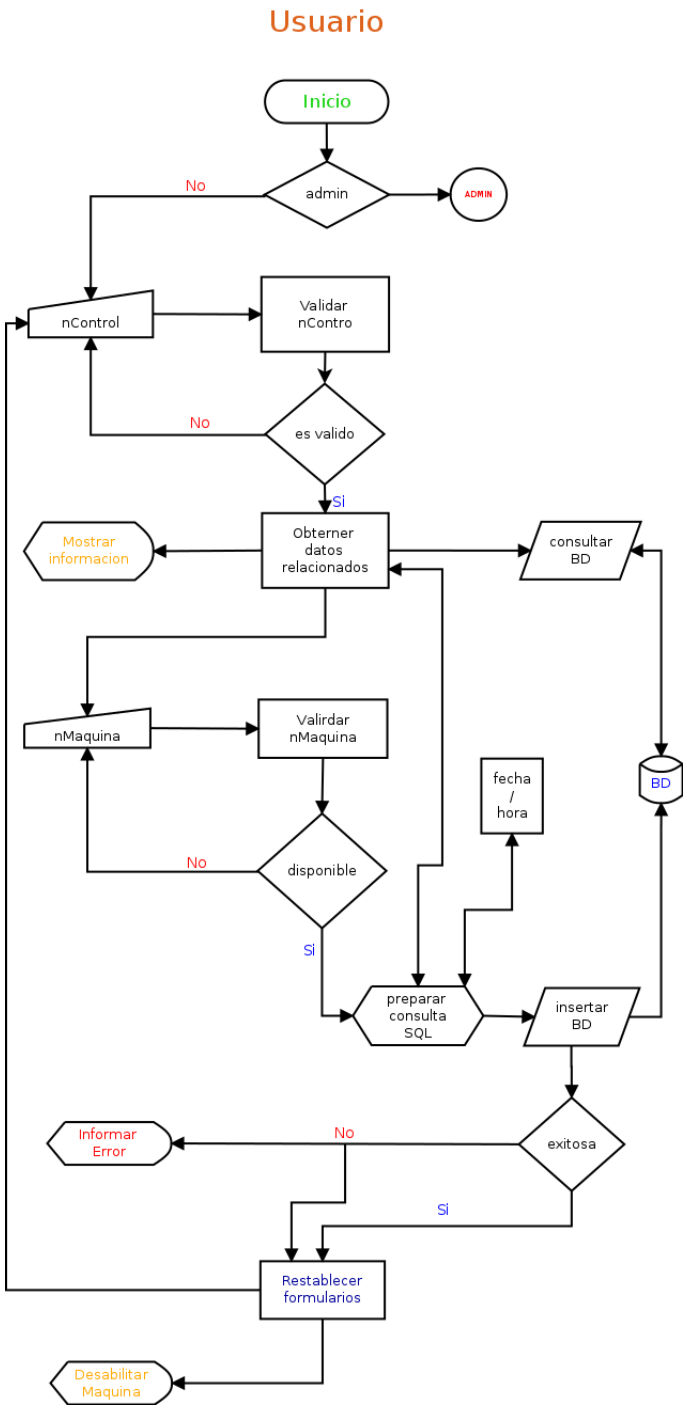

Figura 3: Proceso de registro.

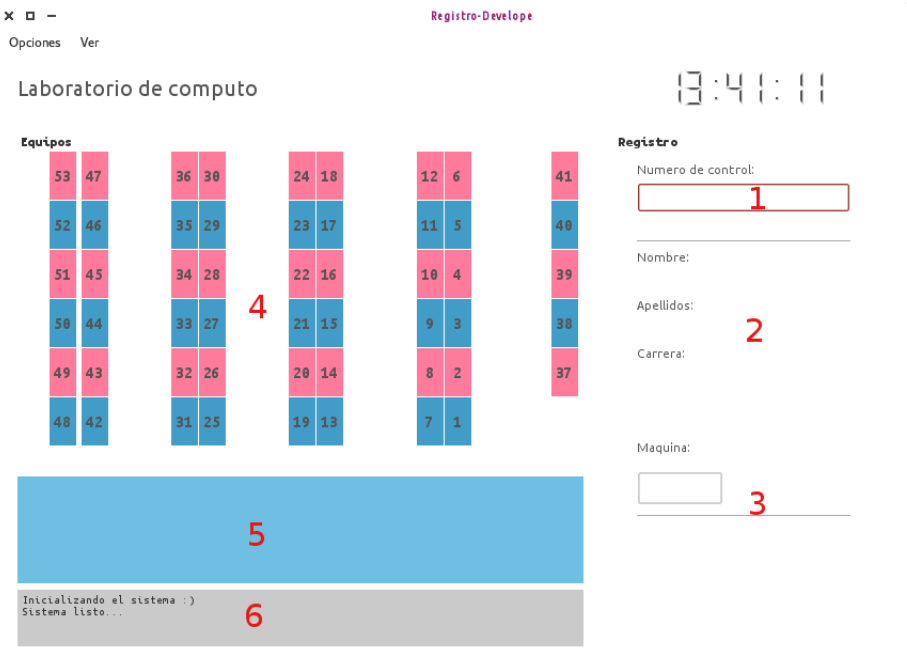

Figura 4: Secciones de la interfaz de usuario.

3.4.3. Diseño de la base de datos

De la mano del núcleo del sistema, fue necesario diseñar una base de datos que permitiera almacenar los datos que se generan, es simple, pero sin ella no seria posible que el sistema cumpla con los objetivos planteados, hasta el momento se compone de tres tablas:

Alumnos almacena los datos de los alumnos activos en el SII.

Entrada almacena los datos de entrada.

Salida para cada entrada ingresa una fecha y hora de salida.

3.5. Implementación

3.5.1. Construcción de la base de datos

La base de datos es la parte central de todo el sistema, pues es donde se almacenaran los datos generados por el sistema, para los fines que después se requieran por ello y seleccionar un manejador de base de datos compatible con las tecnologías usadas es de gran importancia, la opción fue usar `MariaDB`, la cual es en términos simples una version de `MySQL` potenciada y con muchas características añadias que lo hacen ideal para ser el motor de base de datos.

En la imagen 5 podemos ver la estructura de las tablas que componen la base de datos.

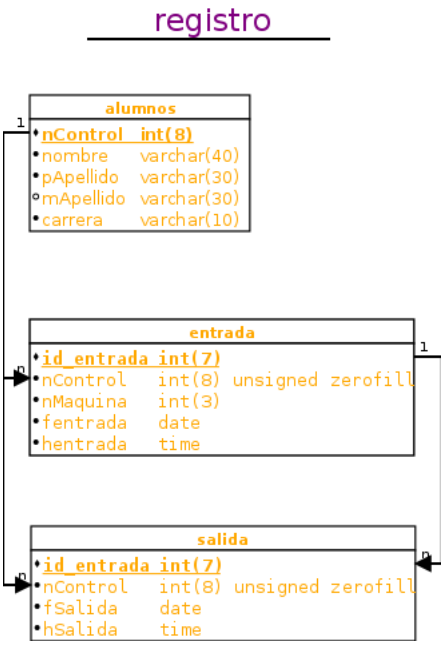

Figura 5: Tablas de la base de datos

3.5.2. Construcción de los módulos de sistema

La metodología elegida exige la cosntruccion de prototipos, lo que es lo mismo, cada uno de los modulos descritos la seccion 3.2 se construyo por separado, con los elementos estrictamente necesarios para su funcionamiento.

Siendo agregados al proyecto solo y únicamente hasta que estuvieron completos y funcionales.

Explicar el funcionamiento interno de cada modulo, escapa de las posibilidades de este documento, mas no significa que la documentación no exista, esta está alojada en el sistema de control de versiones `CVS`, `GitHub`, en el siguiente enlace, el repositorio por el momento esta administrado por Juan Carlos Ávila, aunque se puede ramificar por quien guste.

Sistema para el registro de entradas y salidas de áreas criticas *GitHub* el documento esta perfectamente documentado, en cada una de las funciones y clases que lo componen, por lo que incluir la descripción lógica de los mismos seria redundante, para ello revise la documentación de cada modulo en el repositorio.

3.5.3. Problemas por resolver

En este punto y pese a que el sistema es funcional, aun hay detalles que arreglar, en especial del lado del administrador del mismo, de nuevo cada uno de los elementos que se agreguen, serán debidamente documentados tanto en este documento como en el correspondiente a cada nuevo modulo generado, todo esto alojado en el repositorio del proyecto.

3.6. Pruebas

Las pruebas que se aplicaron al sistema se basan en las pruebas fueron las que recomienda la metodología usada 2.1, son las pruebas unitarias, las cuales son en resumen pruebas aplicadas durante el proceso iterativo del desarrollo, la idea general es desarrollar un módulo o función por separado o con los mínimos elementos para que funcione y una vez que se superan las expectativas esta se integra en el sistema principal en donde se hace la segunda prueba o de integración.

En este caso cada una de las funciones que lo permitían se crearon por separado, y una ves que estuvieron funcionando se integraron con el resto del sistema, de esta forma solo las piezas que en efecto cumplían con su objetivo, se agregaron al conjunto de funciones o definiciones que componen el resto del sistema.

Cada que una nueva funcionalidad se agrego al proyecto, este se guardo como una instantánea del sistema, esto para que los cambios futuros se agreguen a una versión actualizada del mismo.

3.6.1. Resultados de las pruebas

Dada la metodología los resultados de las pruebas ya sean buenas o malas se corrigen en el momento, pues si la nueva funcionalidad que se este construyendo, no supera satisfactoriamente las mismas, esta no puede ser agregada al sistema. En pocas palabras una función no esta lista hasta que no este lista.

3.6.2. Mejoras realizadas después de las pruebas.

Dada la naturaleza de metodología y como ya se menciono anteriormente, las pruebas se implementan en el mismo proceso de desarrollo, con las pruebas unitarias, lo que hace que cuando una nueva función se agrega al sistema, esta ya esta ya a superado las pruebas necesarias, por lo que las pruebas del sistema en general no llegan a darse.

3.6.3. Detalles pendientes por arreglar

3.7. Resultados obtenidos

En este punto de la iteración el sistema esta en una etapa funcional, al menos en lo que se refiere al funcionamiento general, funciona en lo que corresponde al enviar y recibir mensajes, la conexión al sistema y proceso de validación funcionan, aún no se ha implementado una forma de mantener los datos en una base de datos no obstante para el siguiente prototipo sera una de las principales funciones a implementar, justo con un formulario de registro.

El aplicar los pasos de la metodología de desarrollo extremo, brinda resultados en poco tiempo de desarrollo, mientras que si las características cumplen o no con las funcionalidades esperadas, estas se pueden cambiar o mantener para próximas generaciones generando aún así prototipos funcionales en cada iteración.

3.8. Conclusión

Realizar este proyecto ha sido de gran ayuda, tanto profesional como personal, pues cada vez profundizo mas en esas actividades que tanto me apasionan, conocer mas características y posibilidades de los lenguajes y librerías que se usaron para el desarrollo del sistema. Es solo el comienzo, mejores cosas están por venir creo que lo ideal seria construir un sistema genérico que pueda adaptarse a cualquier problemática de registro

de entradas y salidas. Si bien es posible que ya existan sistemas de este tipo, para mi la motivación es más que nada por el carácter didáctico, pues a fin de cuentas estoy en proceso de preparación. Si hay algo que aprendí, es que la disciplina puede hacer las cosas mas fáciles, creo que sera algo que ponga en practica en mis proyectos futuros y en las mejoras que tengo contempladas para este mismo.

Glosario

ITSJR

Instituto Tecnológico de San Juan del Río 4

SII

Sistema Integral de Información 4, 14

Referencias

[1] David Luckham [*The Power of Events - An Introduction to Complex Event Processing in Distributed Enterprise Systems*]. Addison-Wesley, ISBN 0-201-72789-7.

[2] Adolfo Lozano Tello [*Iniciación a la programación utilizando lenguajes visuales orientados a eventos.*] [ISBN978-84-7897-714-7] Ed.Bellisco Ediciones Técnicas y Científicas, ISBN 84-95279-49-5. ISBN 978-84-95279-49-1.