UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Paradigmas de Programación

Informe N°1

Sistema de gestión fastClinic

Nombre:	Joaquín Ignacio Villagra Pacheco
R.U.N.:	18.847.934-0
Carrera:	Ingeniería Civil Informática
Año estimado de egreso:	2018
E-mail:	joaquin.Villagra@usach.cl
Profesor:	José Allende
Ayudante:	Maurcio Rojas

Santiago, Chile

Lunes, 22 de Septiembre de 2014

Índice de contenidos

1 INTRODUCCIÓN

Un recinto clínico requiere generar un sistema de gestión de sus flujos de información. Se posee una base de datos existente, distribuida en diversos archivos de texto y se solicita a un estudiante de ingeniería informática generar un software, escrito en lenguaje de programación C que sea capas de manejar un conjunto definido de funcionalidades. Para ello se le detallan reglas y formatos que deben seguir la compilación, ejecución y entrega de resultados de dicho software.

El presente documento resume el trabajo realizado en el desarrollo del software solicitado, a su vez explica cada funcionalidad implementada y los problemas que surgieron en el desarrollo de las mismas. Este documento se compone de los siguientes capítulos:

- Descripción del problema, sección donde se especifican cada funcionalidad a desarrollar, reglas y formatos exigidos.
- Descripción de solución, sección donde se detalla cada una de las funciones y/o procedimientos desarrollados para cumplir las operatividades exigidas.
- Conclusiones, sección donde se realiza una visión global de lo aprendido, problemas y/o errores presentes en el desarrollo y el cumplimiento de los objetivos planteados.
- Referencias, sección donde se informa cada una de las fuentes de información consultadas.
- Anexo Codigo Fuente.
- Anexo Documentación.

2 DESCRIPCIÓN DEL PROBLEMA

2.1 Motivación

Para el recinto clínico *FastClinic* es de imperiosa necesidad manejar su información de una manera sencilla, rápida y centralizada. Su antiguo sistema informático, sin soporte, ya no cubría sus necesidades y por es por esta razón que se ven en la necesidad de solicitar el desarrollo de un nuevo software, teniendo como base de registros archivos de textos heredados del antiguo sistema.

2.2 Especificaciones del proceso de atención

Para el desarrollo del nuevo sistema, la directiva de *FastClinic* nos ha entregado cierta información relevante, referente al proceso de atención de los pacientes y las relaciones existentes entre los distintos datos que se registran.

1. Un paciente es atendido por uno o más médicos y un médico puede atender múltiples pacientes.

2. Un paciente puede llegar con uno o varios motivos de consultas.

3. Un paciente puede tener uno o varios diagnósticos.

4. Los diagnósticos son efectuados por un médico responsable en una fecha (dd/mm/aaaa) particular.

5. Los pacientes pueden recibir distintos tratamientos recomendados por un doctor. Los tratamientos están normalizados para cada diagnóstico.

6. Los diagnósticos pueden ser tratados con uno o más tratamientos. Los tratamientos pueden ser comunes para uno o más diagnósticos.

7. Los pacientes reciben el alta de un diagnóstico en una fecha (dd/mm/aaaa) particular y bajo la responsabilidad de un médico (que puede ser diferente del médico que hizo el diagnóstico original).

2.3 Funcionalidades solicitadas

Dentro del enunciado del proyecto a desarrollar, se detallan diversas funcionalidades que pueden ser implementadas. No se establece que deben implementarse todas, quedando a criterio del tiempo del estudiante decidir cual programar. En el siguiente capitulo, se especifican las funcionalidades implementadas como parte de la solución. En el caso que se quiera conocer el abanico de funciones que podrían haber sido implementadas, favor dirigirse al enunciado del proyecto.

2.4 Dominio del problema

Por parte del estudiante de ingeniería, existe un control del problema ya que gracias a lo aprendido en cursos anteriores puede aplicar la lógica de resolución de problemas en este proyecto y a su vez, el manejo del lenguaje C no es una falencia, ya que estos contenidos han sido practicados en el curso actual de paradigmas de programación.

3 DESCRIPCIÓN DE LA SOLUCIÓN

3.1 Propósito de la solución

El proposito de nuestra solución, es generar un software de gestión que cumpla los estándares especificados por la gente de FastClinic y que sea capaz de entregar satisfacción al usuario, optimizando el tiempo invertido en los procesos productivos de dicha entidad.

3.2 Funcionalidades implementadas

A continuación se detallan cada una de las funcionalidades trabajadas, escribiendo el detalle de cada funcionalidad en el formato de documentación de Doxygen.

1. @file funciones.c
   @brief obtenerNombrePaciente (Función de nivel bajo)
   @param char runPaciente[]
   @returns none
   @verbatim
   Función que recibe como parámetro el run de un paciente e indica el nombre corrrespondiente al run del paciente ingresado. No posee retorno, ya que posee una salida directa a un archivo de texto.
   @endverbatim

2. @file funciones.c
   @brief especialidad (Función de nivel bajo)
   @param char runDoctor[]
   @returns none
   @verbatim
   Función que recibe como parámetro el run de un médico e indica el nombre correspondiente al run del medico ingresado. No posee retorno, ya que posee una salida directa a un archivo de texto.
   @endverbatim

3. @file funciones.c
   @brief tratamientoRiesgoso (Función de nivel bajo)
   @param char argumento[]
   @returns none
   @verbatim

Función que recibe como parámetro el nivel de riesgo de un tratamiento e indica el nombre corrrespondiente a todos los tratamientos que poseen dicho nivel de riesgo especificado. No posee retorno, ya que posee una salida directa a un archivo de texto.
@endverbatim

4. @file funciones.c
   @brief tratamientoMasUsado (Función de nivel bajo)
   @param int idEntrante
   @returns none
   @verbatim
   Función que recibe como parámetro un identificador, el cual sirve para diferenciar las dos aplicaciones de la función. Si el idEntrante es igual a cero, se realiza la busqueda simple del tratamiento más usado. Si no, se ejecuta la búsqueda del tratamiento más usado para un determinado diagnostico. La segunda segunda sección no esta implementada, quedando solo operativa la búsqueda simple del tratamiento más usado.
   No posee retorno, ya que posee una salida directa a un archivo de texto.
   @endverbatim

5. @file funciones.c
   @brief medicoMasAltas (Función de nivel bajo) @param none
   @returns none
   @verbatim
   Función que obtiene el run, nombre y apellido del Medico que otorga más altas.
   No posee retorno, ya que posee una salida directa a un archivo de texto.
   @endverbatim

6. @file funciones.c
   @brief riesgoUltimoTratamiento (Función de nivel alto)

@param char nombre[], char apellido[]

@returns none

@verbatim

Función que recibe como parámetro el nombre y apellido de un paciente X, y obtiene el riesgo del ultimo tratamiento que este ha recibido. No posee retorno, ya que posee una salida directa a un archivo de texto.

@endverbatim

7. @file funciones.c

   @brief diagnosticoPacienteMedico (Función de nivel alto)

   @param char nombreDiagnostico[], char runDoctor[]

   @returns none

   @verbatim

   Función que recibe como parámetro el nombre del diagnostico especificado y el run del medico diagnosticado. Con dichos datos, realiza una búsqueda de vinculaciones, obteniendo los pacientes diagnosticados con dicho diagnostico X por el médico Y. No posee retorno, ya que posee una salida directa a un archivo de texto.

   @endverbatim

8. @file funciones.c

   @brief listarMedicosTratantesPaciente (Función de nivel alto)

   @param char runPaciente[]

   @returns none

   @verbatim

   Función que recibe como parámetro el run de un paciente en especifico, realiza una busqueda de vinculaciones, obteniendo los datos de los médicos que han atendido al paciente indicado. Se consideran medido de diagnostico y médico que da el alta. No posee retorno, ya que posee una salida directa a un archivo

de texto.

@endverbatim

9. @file funciones.c

   @brief rutPacienteDiagnostico (Función de nivel medio)

   @param char runPaciente[]

   @returns none

   @verbatim

   Función que recibe como parámetro el run de un paciente e indica el diagnostico mas frecuente de dicho paciente. No posee retorno, ya que posee una salida directa a un archivo de texto.

   @endverbatim

10. @file funciones.c

    @brief tratamientosDiagnosticoPaciente (Función de nivel medio)

    @param int idDiagnostico, int idPaciente

    @returns none

    @verbatim

    Función que recibe como parámetro el identificador del diagnostico recibido y el identificador del paciente que recibe dicho tratamiento. Se determina que tratamiento puede recibir el paciente a partir de su diagnostico. No posee retorno, ya que posee una salida directa a un archivo de texto.

    @endverbatim

11. @file funciones.c

    @brief medicosTratantes (Función de nivel medio)

    @param int idPaciente

    @returns none

    @verbatim

    Función que recibe como parametro el identificador de un paciente, realiza

una busqueda en "BD" y entrega la información (nombre-apellido) de todos lo médicos tratantes del paciente especificado. No posee retorno, ya que posee una salida directa a un archivo de texto.

@endverbatim

12. @file funciones.c

    @brief tratamientosPacienteCorreo (Función de nivel medio)

    @param char correo[]

    @returns none

    @verbatim

    Función que recibe como parámetro el correo electrónico de un paciente, realiza una busqueda en "BD" y entrega una lista de todos los tratamientos recibidos por paciente especificado. No posee retorno, ya que posee una salida directa a un archivo de texto.

    @endverbatim

13. @file funciones.c

    @brief eliminarPaciente (Función de nivel medio)

    @param char runPaciente[]

    @returns none

    @verbatim

    Función que recibe como parámetro el run de un paciente, realiza una búsqueda en "BD" y elimina los registros del paciente indicado, siempre y cuando, este paciente no tenga diagnósticos en los registros del sistema. No posee retorno, ya que la operación se refleja en el archivo Paciente.txt.

    @endverbatim

14. @file funciones.c

    @brief eliminarMedico (Función de nivel medio)

    @param char runDoctor[]

@returns none

@verbatim

Función que recibe como parámetro el run de un medico, realiza una busqueda en "BD" y elimina los registros del médico indicado, siempre y cuando, este médico no sea responsable de diagnósticos o tratamientos vigentes. No posee retorno, ya que la operación se refleja en el archivo Doctor.txt

@endverbatim

15. @file funciones.c @brief modificarCorreoPaciente (Función de basico)

    @param char modificarCorreoPaciente[]

    @returns none

    @verbatim

    Función que recibe como parámetro el correo actual de un paciente y el nuevo correo a registrar. Reemplaza el correo antiguo por el nuevo correo ingresado. No posee retorno, ya que la operación realizada se refleja en el archivo Paciente.txt

    @endverbatim

4 ANÁLISIS DE RESULTADOS

En el desarrollo, solo se desarrollaron las funcionalidades que a juicio del estudiante fueran abordables. Es por esta razón que los resultados, teniendo en cuenta unica y exclusivamente las funcionales trabajadas, son muy buenos. Ya que cada funcionalidad desarrollada, logra dar solución a lo que se solicitó por parte de FastClinic.

Quizás, mirándolo desde otra perspectiva, no son los mejores resultados ya que no se logra dar solución a todo el abanico de funcionalidades presentadas, sin embargo se logra cubrir una gran parte de ellas, superando con creces el mínimo exigido.

5 CONCLUSIÓN

Durante el desarrollo del software, se presentaron diversas complicaciones. Por una parte, los errores de redacción del enunciado del proyecto propiciaron confusiones del estudiante en cuestión y generaron retardos en la implementación de las funcionalidades solicitadas. Luego de haber "entendido" lo que se solicitaba, y faltando no más de dos días para la entrega del proyecto, llega información de aclaración de las dudas respecto de estas funcionalidades. Esta información de ultimo minuto genera un trabajo inesperado de corrección y refinamiento de procesos que no estaba contemplado en un principio.

Todas estas complicaciones pudieron ser superadas gracias a la dedicación y el arduo trabajo desempeñado, pero hay que saber aprender de los errores y buscar un mejor manejo de los tiempos, contemplando posibles detalles de ultimo minuto.

En cuanto al lenguaje de programación C, pese a ser un lenguaje antiguo no se queda atrás en poder de procesamiento. Esto puede verse muy bien en los sistemas operativos provenientes de UNIX, porgramados mayoritariamente en C, sistemas robustos y utilizados para tareas de alta demanda como la gestión de un servidor.

Finalmente, se puede visualizar bajo la propia experiencia las diferencias entre los distintos paradigmas de programación, teniendo ahora como comparar objetivamente, en un ambiente real las ventajas y desventajas que nos otorga cada paradigma en cuestión.

6 REFERENCIAS

- Andrés Marzal, Isabel Garcia. (Año no informado). Introducción a la programación con C. Departamento de lenguajes y sistemas informáticos: Universitat Jaume.

- Brian W. Kernighan and Dennis M. Ritchie. (1988). The C programming Language. EEUU: Prentice-Hall.