UCSC | FACULTAD DE INGENIERÍA

Clínica odontológica al estilo C++

Estudiantes: Esteban Cadiz Leyton
Cristian Vazquez Baeza
Daniel Aravena Contreras

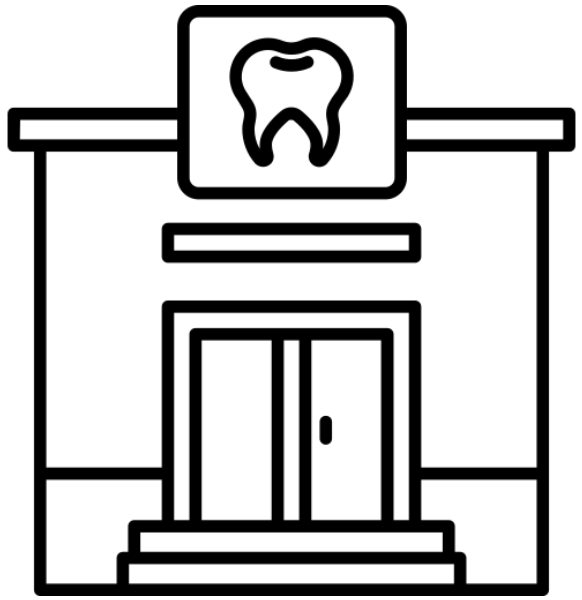
Docentes: Braulio Quiero Hernandez
Hector Ayala Peñailillo

Curso: Estructura de datos

Fecha: 2 de julio de 2024

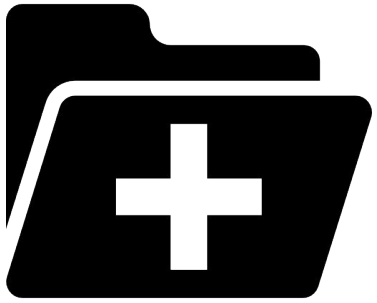
Introducción:

El presente proyecto se ha creado principalmente para solucionar problemas laborales en una clínica odontológica mediante un código de C + +, mejorando la organización entre los pacientes, y facilitando a los miembros del equipo las diversas tareas más importantes a la hora de atenderlos.



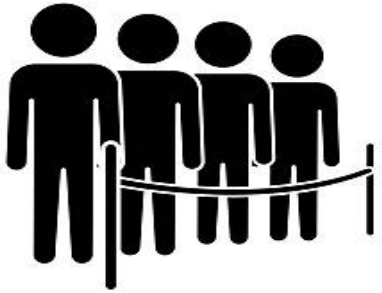
ucsc.cl

Objetivos principales



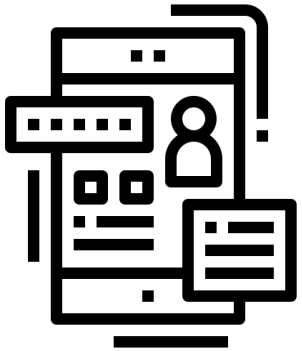
Facilitar el almacenamiento de las fichas

Implementar un método eficaz para el guardado, acceso y modificación a las fichas de los pacientes



Priorizar pacientes

La fila de espera de los pacientes está hecha de tal modo que se atenderán las urgencias en primer lugar, y luego por orden de llegada.



Interfaz sencilla

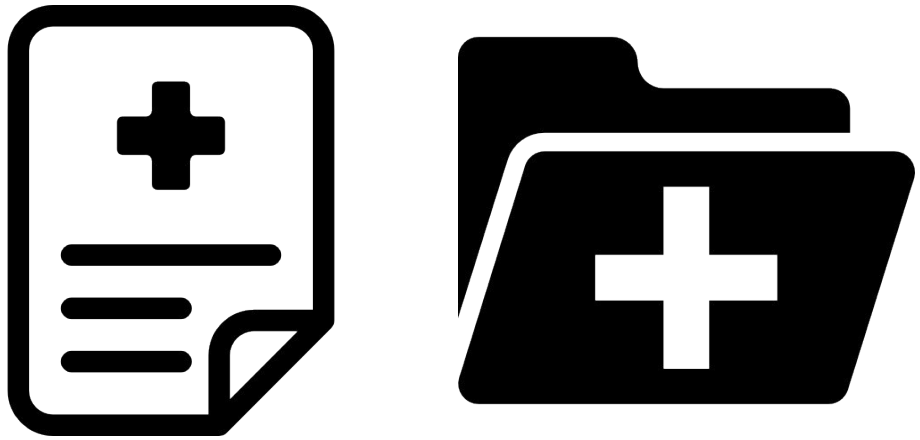
Una interfaz sencilla de entender para el equipo de trabajo de la clínica, y así facilitar su uso diario.

ucsc.cl

Definición del Problema

Organización al llegar por prioridad

Evitar conflictos entre pacientes, se añadirá a la fila por orden de llegada, y se les dará prioridad a aquellos con urgencia

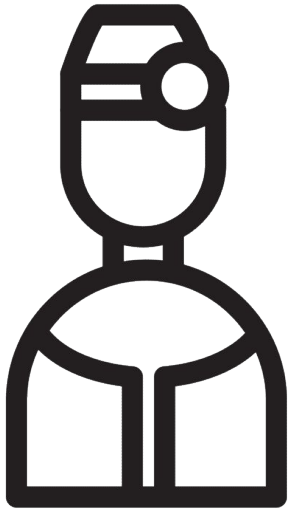


Manipulación de fichas

Agregar, modificar o quitar fichas de manera eficiente y sencilla para las personas de secretaría, para así tener mejor control de las mismas

ucsc.cl

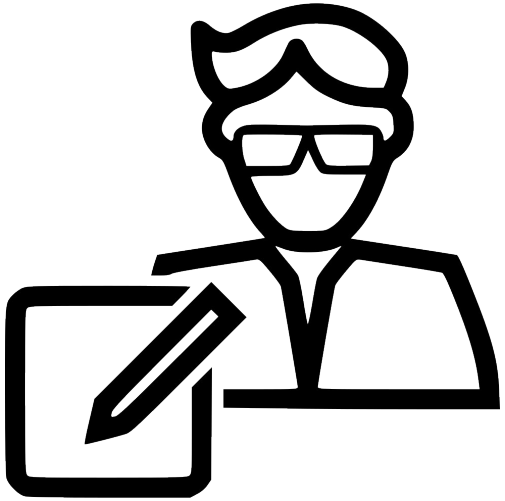
Interesados



Dentista

Personas que revisan a los pacientes y redactan su ficha dental con sus necesidades. Necesitan revisar al paciente para atender que problema dental específico.

Su interés es tomar el registro de lo que le sucede al paciente.



Secretaría

Personas que agendan horas de pacientes, sus prioridades que modifican sus fichas con orden del dentista encargado.

Su interés es tener una forma eficaz y ordenada de guardar los datos de los pacientes.

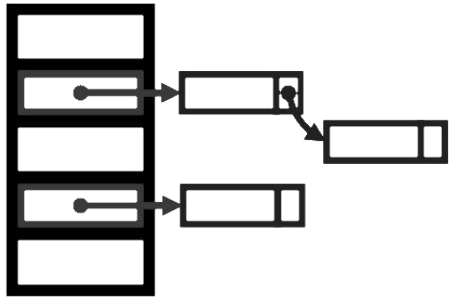
Pacientes

Personas que necesitan pedir una hora dental, ya sea de urgencia o no.

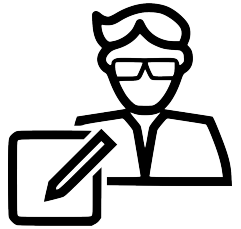
Sus intereses son ser atendidos de forma rápida y simple, tanto por la secretaría, como por los dentistas.

Funciones del Producto

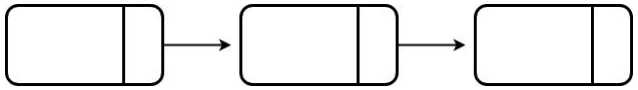
Módulo 1: Tabla Hash



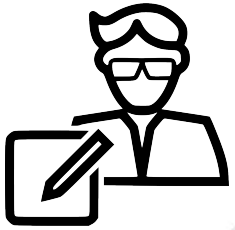
- Consultar ficha pacientes: Dentistas y Secretaría
- Manipular ficha pacientes: Dentistas y Secretaría

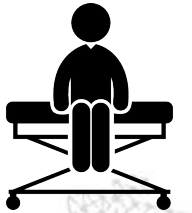


Módulo 2: Fila



- Asignar prioridad al llegar: Secretaría y Pacientes
- Consultar próximo turno: Secretaría y Pacientes





ucsc.cl

Elaboración

Diagramas hechos en Draw.io

Diagrama 1: Tabla Hash

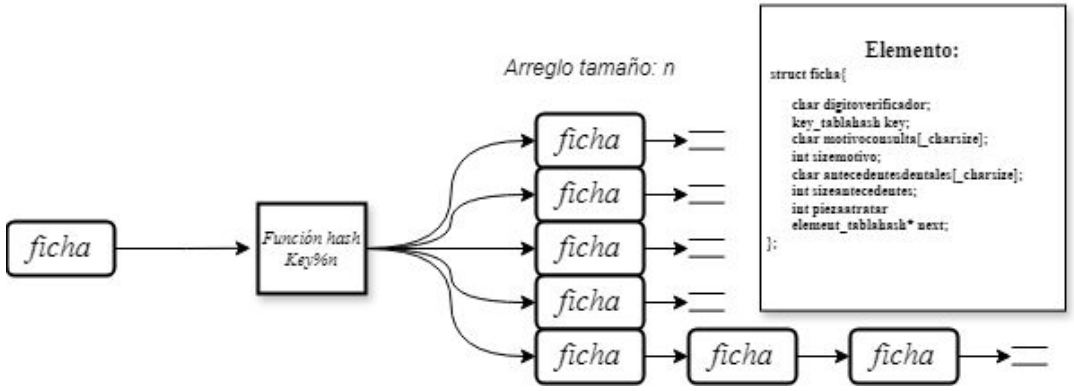
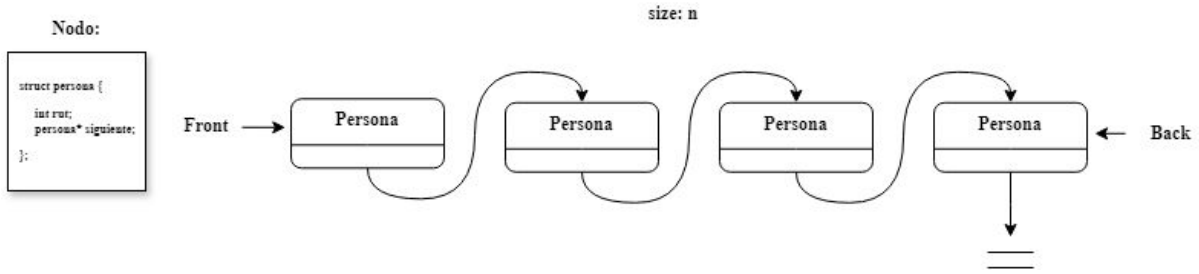


Diagrama 2: Fila



ucsc.cl

Construcción

```
1  void ingresofilaespera(ListQueue *urgencia, ListQueue *normal) // funcion que añade a alguien a la fila de espera
2  {
3      elemento_fila rut;
4      int ingresofila;
5
6      std::cout << "Ingrese rut paciente" << "\n";
7
8      std::cin >> rut;
9      fflush(stdin); // limpia buffer
10
11     std::cout << "Ingrese a que fila lo desea enviar" << "\n";
12
13     std::cout << "1. Atencion Normal" << "\n";
14     std::cout << "2. Atencion de Urgencia" << "\n";
15
16     std::cin >> ingresofila;
17     fflush(stdin); // limpia buffer
18     if (ingresofila == 1)
19     {
20         (*normal).enqueue(rut);
21     }
22     else
23         (*urgencia).enqueue(rut);
24 }
```

Código 1: Ingreso a fila de espera

```
1  void consultaficha(HashEncadenado *fichasguardadas) // funcion para consultar fichas
2  {
3
4      element_tablahash fichaaux; // ficha auxiliar
5      key_tablahash rutk;
6      std::cout << "Ingrese rut sin digito verificador ni puntos " << "\n";
7      std::cin >> rutk;
8      fflush(stdin); // limpia buffer
9      fichaaux = (*fichasguardadas).find(rutk);
10     if (fichaaux.key != 0)
11     { // caso donde hay ficha
12         printficha(fichaaux);
13     }
14     else
15     { // Caso donde no encuentra ficha es decir find devuelve key=0
16         std::cout << "No se encontro ficha" << "\n";
17     }
18 }
```

Código 2: Consulta ficha en tabla hash

Principales dificultades

- Clase std::string y su incompatibilidad con “malloc()”
- Necesidad de pasar por referencia al usar funciones en “main”
- Requerimiento de “fflush(stdin)” función de C

Conclusión

Los tres objetivos principales fueron cumplidos de forma exitosa. Como requisitos a futuro, queremos crear una interfaz gráfica, para que sea mucho más amigable para el usuario.

Abordamos mucho sobre más contenidos que no logramos ver en clases, como la Tabla Hash, al igual que buscar soluciones a problemas por diferentes medios.