# **Pitch Proyecto – Primer Corte**

**Introducción**¿Alguna vez has esperado horas en un hospital, preguntándote por qué el sistema parece tan caótico?

Los hospitales enfrentan colas interminables y una gestión ineficiente de personal y ambulancias. Esto genera demoras, pacientes mal priorizados y recursos desaprovechados, afectando la calidad de la atención y la experiencia de todos.

El objetivo de este proyecto es optimizar la gestión hospitalaria mediante un sistema de software que reduzca los tiempos de espera, priorice adecuadamente a los pacientes y gestione de forma eficiente al personal médico y las ambulancias.

**Problema y Motivación** En nuestro análisis identificamos tres problemáticas principales:

1. **Pacientes en salas de espera** durante mucho tiempo debido a la priorización de urgencias y la limitada cantidad de salas.
2. **Ausencia de un sistema objetivo de triage**, lo que provoca que casos leves esperen horas innecesariamente.
3. **Gestión ineficiente del personal y ambulancias**, dificultando la asignación rápida de citas y recursos.

Esto nos llevó a plantear una solución de software **mantenible, escalable y confiable**, capaz de adaptarse a las necesidades de diferentes hospitales.

**Enfoque de Ingeniería de Software** Para el diseño seguimos los fundamentos de ingeniería de software, aplicando **principios SOLID** para garantizar bajo acoplamiento y alta cohesión entre módulos.  
 Dividimos la solución en módulos:

* **Pacientes y Triage** (gestión de colas y nivel de urgencia),
* **Personal Médico y Citas**,
* **Ambulancias y Recursos de Emergencia**.

Esto nos permitirá que el sistema pueda evolucionar fácilmente con nuevas reglas de prioridad o servicios.

**Arquitectura** Nuestra solución se diseñó bajo un estilo **en capas**, ya que este se ajusta a las necesidades del proyecto en esta primera fase.  
 Con este enfoque logramos una **separación clara entre la lógica de negocio, la gestión de eventos y la interacción con el usuario**, facilitando la escalabilidad y la simplicidad de implementación.  
 Además, utilizamos patrones como **Strategy** para el cálculo de triage y **Observer** para la notificación de eventos importantes (citas asignadas, ambulancias despachadas).

**Pruebas y Calidad** Adoptamos la práctica de **TDD (Test-Driven Development)** para las funciones críticas, como el cálculo de triage y la asignación de pacientes.  
 Esto nos permitió obtener un **código más confiable, limpio y preparado para refactorizaciones**, garantizando calidad desde las primeras etapas.

**Cierre** En conclusión, en este primer corte logramos:

* Definir la arquitectura base de la solución.
* Implementar un prototipo funcional en Python con CLI que gestiona pacientes, triage, citas y ambulancias.
* Documentar la arquitectura con principios SOLID y patrones de diseño.