|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RAPPI – DISEÑO PRUEBA BACKEND  |  | | --- | |  | | [MAURICIO.BEDOYA@GMAIL.COM](mailto:MAURICIO.BEDOYA@GMAIL.COM) | |  | | 3004802278 | | MEDELLÍN | |  | |  | | --- | | ELABORADO POR: mAURICIO BEDOYA LONDOÑO Objetivo: El objetivo de este documento es la conceptualización del diseño de la prueba de backend como proceso de selección para Rappi. | |

LISTADO DE VERSIONES

|  |  |  |  |
| --- | --- | --- | --- |
| Descripción | Versión | Fecha | Creado por |
| Versión inicial que muestra cómo se implementará la solución y sirve para retroalimentación temprana por parte de Rappi. | 1.0 | 2017/03/14 | Mauricio Bedoya |
|  |  |  |  |
|  |  |  |  |

## LISTADO DE CONTENIDO

[DEFINICIONES DE ARQUITECTURA 4](#_Toc477369939)

[a. Estilo Arquitectónico 4](#_Toc477369940)

[b. Patrones de diseño 4](#_Toc477369941)

[c. Rationale 4](#_Toc477369942)

[DISEÑO DE ALTO NIVEL 6](#_Toc477369943)

[DISEÑOS DETALLADOS 8](#_Toc477369944)

[OTROS 11](#_Toc477369945)

# DEFINICIONES DE ARQUITECTURA

* 1. Estilo Arquitectónico: Tanto los patrones de Arquitectura como los estilos Arquitectónicos dependen de 3 factores:
     1. Requerimientos funcionales
     2. Restricciones técnicas
     3. Atributos de calidad

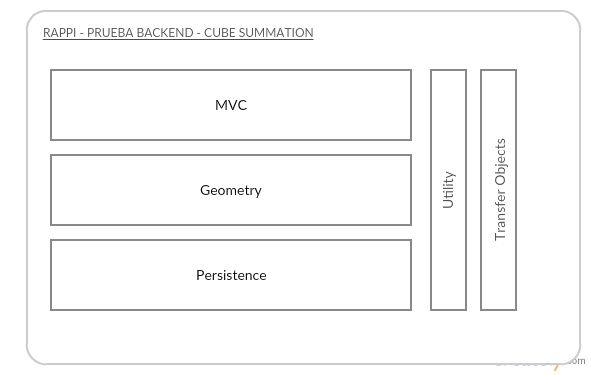
Para el caso específico del test backend para Rappi, sólo se tienen requerimientos funcionales, no hay restricciones técnicas o atributos de calidad, así que la decisión para esta prueba sería usar un modelo vista controlador (MVC) con Angular JS y almacenamiento local (SQLite), un modelo por capas no sería necesario ya que esta no es una solución que se debe mantener en el futuro ni tampoco extender, sin embargo, y como es requisito de la prueba como elemento clave el manejo de patrones de arquitectura, se combinarán diversos patrones para demostrar su uso y no porque sean primordiales para la solución específica del problema planteado.

* 1. Patrones de diseño:
     1. Se usará Factory para crear los objetos de Geometría y Persistencia, y así desacoplar cada capa de la solución.
  2. Rationale:

|  |  |  |
| --- | --- | --- |
| Decisión | Sustentación | Tomada por |
| Se implementarán los patrones arquitectónicos Capas y MVC | Se requiere demostrar la capacidad de uso de patrones como parte clave en el proceso de selección | Mauricio Bedoya |
| MVC no estándar en .Net | Se usará un MVC en el cual la vista no usará elementos .Net (razor y otros), será una vista con Html, Javascript, Css y Bootstrap más estándar y totalmente desacoplado de .Net, esto para que la vista en caso de ser necesaria portarla a otro backend y para que el backend soporte cualquier otro cliente. Se harán llamados HTTP Post al .Net para la interacción cliente-servidor. | Mauricio Bedoya |
| Se usará .Net para backend | No hay restricción y es el lenguaje en el cual se tiene mayor experiencia | Mauricio Bedoya |
| Se usará MySql para persistencia | No hay restricción y es el motor con el que se cuenta en el servidor que uso en la nube, así que es más sencillo el despliegue. | Mauricio Bedoya |
| No se usará el modelo de ingreso de datos sugerido en hackerrank:  2  4 5  UPDATE 2 2 2 4  QUERY 1 1 1 3 3 3  UPDATE 1 1 1 23  QUERY 2 2 2 4 4 4  QUERY 1 1 1 3 3 3  2 4  UPDATE 2 2 2 1  QUERY 1 1 1 1 1 1  QUERY 1 1 1 2 2 2  QUERY 2 2 2 2 2 2  Se usará una interfaz web que permita una mejor manipulación de los datos del cubo. | Mejora en la experiencia de uso de la aplicación. | Mauricio Bedoya  (En espera de aprobación por parte de Rappi) |

# DISEÑO DE ALTO NIVEL

* 1. Capas: Para la implementación de la solución se usarán las siguientes capas:



MVC: Permite separar responsabilidades entre capa de interacción con el usuario y otras capas de aplicación, permite que la interfaz y el ciclo de vida de la página sean más sencillos.

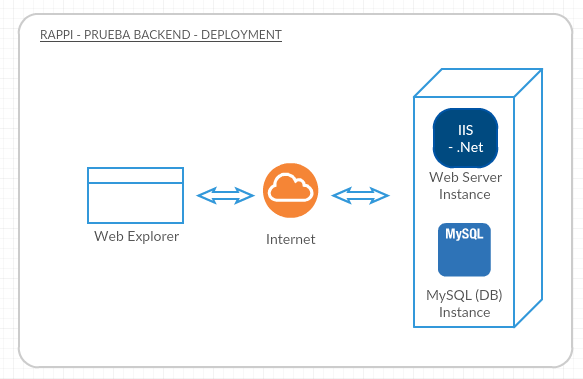
Geometry: Esta capa es la encargada de realizar las lógica de operaciones con figuras geométricas, para este caso el Cubo, pero se pueden adicionar otras figuras.

Persistencia: Esta capa es la encargada de garantizar que la información del cubo se mantenga sin importar si se ha perdido sesión o no (se almacena un único cubo), cabe resaltar que para este ejercicio no se manejará información independiente por cada usuario, TODOS los usuarios que ingresen al sitio verán la misma información (todos comparten un único cubo), ya que para manejar información única se deberían crear componentes de seguridad para inicio de sesión registro, etc, lo cual está por fuera de este alcance.

Utility; Se manejará una capa en donde se tengan funciones generales y comunes que pueden ser usadas por las demás capas. En este caso puntual sólo se usa para el registro de errores de aplicación.

Transfer Objects: Esta capa se usa para definir los objetos que viajan entre capas del Backend.

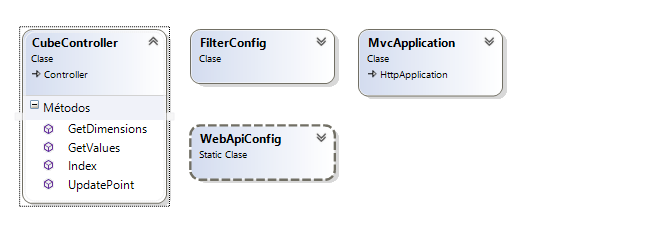
* 1. Despliegue: A continuación se describe como se despliega la solución



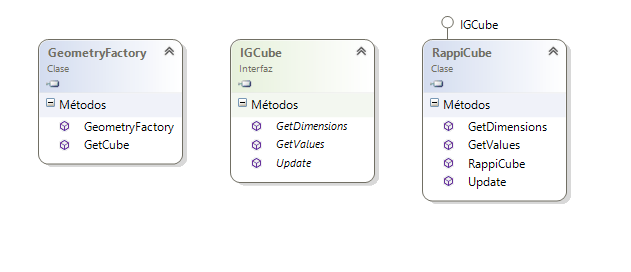
La aplicación será desplegada en Amazon EC2 instances, el servidor web y la base de datos estarán en el mismo servidor, para efectos de la prueba y hacerlo simple, no se pretende ilustrar que esta sea una buena práctica o no.

# DISEÑOS DETALLADOS

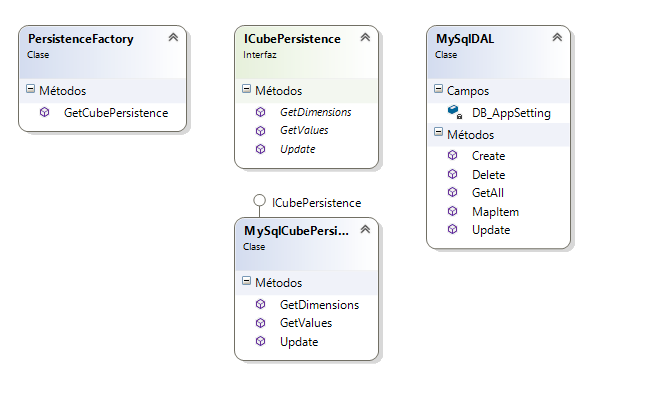
MVC



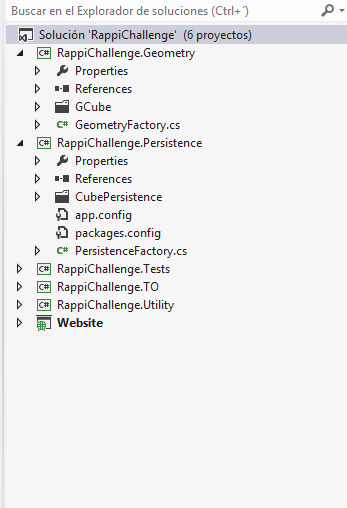
Geometry



Persistence



Distribución de Proyectos en Visual Studio



# OTROS

* 1. Control de código fuente: Se usará github para controlar los fuentes y documentación del proyecto.
  2. Interfaz: Esta es la interfaz gráfica usada

