<Nomenclatura del Proyecto>

Modelo de Arquitectura de Soluciones Técnicas

Versión 1.0

30/04/2020

Histórico de Versiones

| **Fecha** | **Versión** | **Descripción** | **Autor** |
| --- | --- | --- | --- |
| 01/05/2020 | 1.0 |  | Yonamine A. |

Tabla de Contenido

[1. Introducción 4](#_Toc39207851)

[1.1 Objetivo 4](#_Toc39207852)

[1.2 Definiciones, acrónimos y abreviaturas 4](#_Toc39207853)

[1.3 Referencias 4](#_Toc39207854)

[2. Descripción general de la solución 4](#_Toc39207855)

[3. Criterios técnicos 4](#_Toc39207856)

[4. Requerimientos específicos de arquitectura 4](#_Toc39207857)

[5. Arquitecturas de solución viables 4](#_Toc39207858)

[6. Firmas de elaboración, revisión y aprobación 5](#_Toc39207859)

**Modelo de Arquitectura de Soluciones Técnicas**

# Introducción

El proyecto de “Cachimbo a Crack” es un impulso de BASE2 S.A., que tiene como público objetivo a los alumnos que buscan mejorar su rendimiento académico y para aquellos que desean impartir su conocimiento. La meta es una aplicación web, de alumnos para alumnos, donde se podrá publicar citas académicas de los cursos que se imparten en la Universidad Tecnológica del Perú. Es por ello, que el presente documento fue redactado para exponer los procesos a seguir para el diseño de la arquitectura del software, así como, los elementos que guiarán la construcción y evolución del aplicativo web a implementar en la institución anteriormente mencionada, como de igual manera, ayudara a guiar el proceso de toma de decisiones durante la implementación del aplicativo.

## Objetivo

El objetivo principal de este documento es orientar a los desarrolladores del proyecto "Cachimbo a Crack", redactando de forma explícita, los lineamentos a seguir, en términos de arquitectura y soluciones técnicas, definidos por Elliot G. (especialista de TI), con el fin de crear una guiar y exponer las consideraciones necesarias para situaciones que requieran tomar decisiones relacionadas con la evolución del aplicativo.

Es presente archivo busca servir como una guía metodológica para el entendimiento y ejecución de iteraciones del proceso de Arquitectura de Solución, convirtiéndose en un insumo fundamental de la estructuración y desarrollo del programa, siendo también, un instrumento para materializar los proyectos de mejoras futuras en cuento a arquitectura, o en el desarrollo de procesos de Planeación Estratégica de TI.

## Definiciones, acrónimos y abreviaturas

* **TI (tecnología de la información):** Se refiere al uso de equipos de telecomunicaciones y computadoras, para la transmisión, el procesamiento y el almacenamiento de datos.
* **Aplicación web:** Herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de internet o de una intranet mediante un navegador.
* **Arquitectura:** Indican la estructura, funcionamiento e interacción entre las partes del software.

## Referencias

El Presente documento se basa en el contenido de los siguientes documentos:

# Descripción general de la solución

Se desarrollarán dos formas de proporcionar citas académicas: enseñanza y aprendizaje. Para publicar una cita, los estudiantes deben registrarse en el formulario de solicitud, dependiendo de si el estudiante está enseñando o estudiando. Para enseñar, debe asignar un curso a su dominio. Esto será evaluado por otro estudiante en su cita académica. el que quiere aprender solo deberá buscar en las citas de enseñanza ya publicadas e inscribirse o publicar una cita de aprendizaje en la cual un alumno que domine el curso se inscribirá y procederá la cita académica.

**2.1 Actores:**

|  |  |  |
| --- | --- | --- |
| **Nombre del actor** | **Descripción** | **Rol dentro del proceso** |
| Alumno | Estudiantes de la Universidad Tecnológica el Perú (UTP). Aquellos que todavía no emplean el aplicativo con una intensión definida. | Ingreso, registro de usuario en el aplicativo e inicio de sesión. |
| Aprendiz | Denominación que se les da a aquellos estudiantes que usan el aplicativo con el fin de buscar asistencia académica | Uso de la herramienta de búsqueda para encontrar reuniones académicas ofrecidas por los “instructores” para una determinada área del conocimiento y curso, creación de un horario de aprendizaje y generación de citas académicas. |
| Instructor | Forma en la que se definen dentro del proceso a aquellos alumnos interesados en impartir clases. | Emplean la herramienta de búsqueda para ubicar los horarios de aprendizaje definidos por el “aprendiz”, creación de horarios de clases dispuestas a impartir y selección de área y materia, además, tendrá la capacidad de generar citas académicas. |

2.2 Actividades:

|  |  |  |
| --- | --- | --- |
| **ID** | **Actividad** | **Descripción** |
| 1 | Registro | Acción por la cual los “alumnos” registran sus datos en la aplicación a fin de poder acceder como usuarios. |
| 2 | Inicio de sesión | Por medio de las credenciales registradas en sistema, los “alumnos” podrán acceder al aplicativo. |
| 3 | Publicar horarios de enseñanza | Los “alumnos” que deseen enseñar, publicaran el horario en el que estan dispuestos a impartir su clase, en este punto, pasan a ser considerados un actor “instructor” |
| 4 | Buscar horarios de enseñanza | Los “alumnos” que cuyo objetivo sea aprender, emplearan la opción de búsqueda integrada en el aplicativo, para ubicar un harario que les convenga, en este punto, son considerado actores “Aprendiz”. |
| 5 | Publicar horarios de aprendizaje | El “Aprendiz”, podra crear horarios acordes a cuando puede y quisiera tomar clases de refuerzo. |
| 6 | Buscar horarios de aprendizaje | El “Instructor” buscara los horarios registrados en el sistema por el “Aprendiz”. |
| 7 | Generar cita académica | Tanto “Aprendiz” como “Instructor”, emplean este mecanismo para registrase a una sesión de aprendizaje, sea para asistir a una clase pautada, o para atender a quien desea aprender. |

# Criterios técnicos

Los criterios técnicos utilizados incluyen los elementos siguientes:

* Costos de desarrollo, implementación y calidad
* Tiempo
* Disponibilidad
* Confidencialidad
* Mantenibilidad
* Soporte de carga simultánea
* Tolerancia a fallos

# Requerimientos específicos de arquitectura

Los requisitos que se deben tener en cuenta al diseñar la arquitectura incluyen:

* REQS\_2: El sistema debe presentar un mecanismo de entrada de información para que el usuario interactúe correctamente con el aplicativo.
* REQS\_3: El sistema debe presentar un mecanismo de salida de información para que el usuario interactúe correctamente con el aplicativo.
* REQS\_4: El sistema debe presentar un mecanismo mediante el cual el administrador pueda realizar cambios en el mismo.
* REQS\_5: El sistema debe presentar un mecanismo para mostrar la información de citas dentro de un curso.
* REQS\_6: El sistema debe presentar un mecanismo para contrarrestar un acceso no autorizado.
* REQS\_7: El sistema debe presentar un mecanismo que no permita la aglomeración de citas desfasadas.

# Arquitecturas de solución viables //Avanzando: Josué Mar//recuerda el formato al terminar bro, arial 10 ~~Claro, los ajusto en un segundo 😊~~ Listo!

**Descripción**

* Arquitectura cliente-servidor: La arquitectura propuesta separa el programa en tres niveles distintos, detallados a continuación.
  + Cliente: El nivel de cliente permite a los usuarios interactuar con el sistema. Los clientes envían mensajes al servidor de aplicación, el cual los procesa y devuelve las respuestas.
  + Servidor de Aplicación: Este servidor aplica la lógica de negocio a los mensajes recibidos de los clientes – de ser necesario comunicándose con el servidor de base de datos para obtener la información necesaria para hacerlo.
  + Servidor de Base de Datos: Provee al servidor de aplicación acceso a las bases de datos usadas en el sistema.
* Arquitectura Publish/Subscribe: La arquitectura propuesta está estructurada de la siguiente manera:
  + Publisher: Permite a los usuarios interactuar con el sistema. Las solicitudes al servidor se envían mediante un canal predefinido según el tipo de usuario (visitante, alumno o administrador) y tipo de mensaje, y son recibidas por la Nube Pub/Sub.
  + Nube Pub/Sub: Recibe los mensajes emitidos por el publisher. Es un canal intermedio entre los Publishers y Subscribers, organizando los mensajes enviados por los Publishers en tópicos y canales de subscripción. Una vez realizado ento, reenvía estos mensajes a los Subscribers que han pedido recibir información de esos canales.
  + Subscribers: Incluye a los servidores, y recibe los mensajes de los canales a los que está suscrito. Cada subscriber solicita a la nube los mensajes que pueden procesar.

**Justificación y beneficios**

* Arquitectura cliente-servidor:
  + Simplicidad: La arquitectura cliente-servidor propuesta únicamente requiere de tres niveles de funcionamiento claramente demarcados y con modos de comunicación predecibles, que hace aumenta su mantenibilidad y reduce los costos iniciales – implementar el sistema con esta arquitectura, como consecuencia de su simplicidad, permitiría más fácilmente construirlo, probarlo e implementarlo.
  + Integración de servicios: La arquitectura cliente-servidor permite compartir recursos, que para a la UTP le permitiría fácilmente realizar acciones como conseguir estadísticas y logs de la información que transita a lo largo de su red.
* Arquitectura Publish/Subscribe:
  + Mínimo acoplamiento: A diferencia de la arquitectura cliente-servidor, en un sistema Pub/Sub los publishers y subscribers no necesitan conocer la topología del sistema – cada uno puede seguir funcionando independientemente de los otros. Esto permite mayor modularidad en el sistema, podiéndose cambiar parcialmente o completamente la arquitectura de los servidores sin afectar la experiencia de los clientes.
  + Escalabilidad: Como consecuencia del mínimo acoplamiento mencionado anteriormente, los sistemas Pub/Sub son en la mayoría de casos más escalables que los basados en la arquitectura cliente-servidor. Los publishers únicamente generan el contenido: como cada paso se realiza a través de un intermediario, únicamente una capa requiere configuración extra para agregar nuevos servidores o usuarios, que para el caso de uso de nuestro cliente le permitiría reducir sus costos de operación.

**Desventajas**

* Arquitectura cliente servidor:
  + Posible sobrecarga de servidores: De haber demasiado frecuentes solicitudes simultáneas por parte de los clientes, los servidores corren el riesgo de sobrecarga severa, formando congestión de tráfico, y disminuyendo la satisfacción de los usuarios del sistema.
* Arquitectura Publish/Subscribe:
  + Inabilidad de coordinar remitentes y receptores: El problema causado por el mínimo acoplamiento entre los publishers y subscribers es que los segundos no pueden saber cuándo los primeros se han conectado correctamente, tanto en las conexiones iniciales como en las reconexiones después de fallas en la red. No existe manera tampoco de acomodar diferentes velocidades de transmisión de datos o solicitar reenvíos de información perdida: los suscriptores deben mantenerse disponibles o arriesgar perder mensajes.

**Otras consideraciones**

* Arquitectura cliente-servidor:
  + La empresa Base2 S.A. posee herramientas para la implementación rápida y efectiva de este tipo de arquitectura, disminuyendo el costo y tiempo iniciales requeridos para el proyecto.
  + Esta arquitectura usa únicamente componentes existentes o de tipo open-source.
* Arquitectura Publish/Subscribe:
  + Esta arquitectura usa únicamente componentes existentes o de tipo open-source.

# Firmas de elaboración, revisión y aprobación

| **Elaboró** | **Revisó** | **VoBo** |
| --- | --- | --- |
| Elliot G.  Especialista de TI | Yonamine A.  Jefe de proyecto | José G.  Analista Funcional |