Development Lead- Prueba Técnica

Estimado(a) candidato(a).

Nos enorgullece saber que Double V Partners tiene el potencial de ser parte de tu proyecto de vida. Sabemos que una prueba técnica es desgastante y algo pesada, pero creemos que un buen proceso de selección es el que nos permite ver el potencial de una persona y también le permite al candidato(a) saber si este va a ser un trabajo retador, divertido, apasionante y acorde a tus expectativas. Date la oportunidad de hacer esto a conciencia no para pasar una prueba, sino para entender si este es el lugar adecuado para tu crecimiento profesional.

La siguiente es una prueba técnica donde tendrás completa libertad para proponer, idear, planear y plasmar tu filosofía de trabajo.

A continuación encuentras una prueba que busca identificar y explorar de forma integral tus conocimientos técnicos, capacidades, habilidades de liderazgo y experiencia. La idea es que generes tu respuesta en un repositorio de github organizando toda la documentación en carpetas teniendo en cuenta las diferentes áreas en las que cae tu rol. Buscamos determinar sobre todo el conocimiento del rol y la forma en que resuelves problemas y avanzas.

Para responder la prueba debes usar un repositorio de GitHub. Envíanos tu respuesta al correo: deysi.perez@doublevpartners.com con el asunto "Prueba Técnica Dev Lead + "_" + tu nombre".¡

Prueba Técnica:

La compañía ha sido contratada para realizar un proyecto en el cual se plantea el desarrollo de un producto digital que debe ser implementado en conjunto con el equipo de producto. Para el desarrollo del producto te han informado que el producto a liberar contempla varias funcionalidades. Este será un producto enfocado en usuario masivo que a partir de la solución podrá:

1. Acceder a un marketplace de eventos artísticos (musicales, conciertos, fiestas) que son promovidos desde redes sociales pero que invitan al usuario a adquirir sus tickets en línea usando la aplicación de la compañía.
2. El usuario podrá comprar boletos usando algún medio de pago (tarjeta de crédito, tarjeta débito).
3. La compañía pretende conocer a sus usuarios y sus preferencias, con lo cual necesita poder generar campañas enfocadas en las preferencias de sus usuarios.
4. La compañía sigue un proceso de publicación de los eventos en el que intervienen varias áreas:

    a) En particular la publicación de un evento comienza por un acuerdo comercial con el artista.

    b) Sobre este acuerdo se generan todos los copies de las campañas, la descripción del evento, los espacios disponibles, el número de cupos, los valores de los tiquetes, fechas de apertura, fecha de cierre de ventas, promociones, códigos de redención, descuentos, productos relacionados.

    c) Luego se generan unas propuestas de cuál debería ser la publicación más adecuada, la cual es aprobada por un equipo de publicación. Cada propuesta puede ser vista por el equipo y aprobada o puede solicitar cambios específicos al equipo que genera la publicación.

    d) Una vez aprobada la publicación esta es publicada en un marketplace.

    e) Para cada producto o promoción disponible, el sistema genera estadísticas de uso, visualización, permite recibir comentarios.

    f) El sistema debe proveer un dashboard de visualización de las estadísticas de cada producto o evento ofrecido.

    g) El sistema debe permitir persistir los carritos de compra en los cuales el usuario realiza el proceso de compra y permitir al administrador conocer las estadísticas de churn del carrito para establecer mecanismos de retoma del proceso.

h) El usuario tendrá la posibilidad de comprar sus tickets sin necesidad de crear cuentas en el sistema, pero podrá registrarse y contar con una cuenta que le permite acceder a beneficios.

i) Una vez realizada la compra el sistema genera un recibo y un tiquete con una clave que debe ser configurada por el usuario de forma que pueda descargar sus tiquetes o visualizarlos entregando dicha clave y un código de compra.

Como Dev Lead debe:

1. Describir la estrategia de desarrollo del proyecto. Establecer un mecanismo de estimación del esfuerzo y plantear un modelo a seguir para el desarrollo.

RTA/
La estrategia de desarrollo se realizará bajo en modelo SCRUM el cual permite desglosar y diagramar con mucho detalle las Historias de Usuario que comprenden la totalidad del desarrollo, dicho modelo permitirá un desarrollo más ordenado y orientado a cumplir con todos los criterios de aceptación dentro de los tiempos establecidos en cada sprint.

Con base en el análisis realizado a los requerimientos para la creación de esta solución tecnológica, se plantea la siguiente propuesta para realizar los desarrollos:

a. Análisis y Diseño. (4 semanas)

- **Requisitos Detallados:** Realizar un análisis exhaustivo de los requisitos funcionales y no funcionales del sistema, priorizándolos según su importancia.
- **Diseño de la Arquitectura**: Definir la arquitectura de microservicios y las interfaces entre ellos. Diseñar la base de datos y la estructura de almacenamiento.
- **Prototipo:** Crear un prototipo interactivo para validar conceptos clave con los usuarios y el equipo de desarrollo.
- **Planificación de Proyecto:** Establecer un plan de desarrollo detallado con cronograma, asignación de recursos y presupuesto preliminar.

b. Desarrollo: (16 semanas)

- Desarrollo de Microservicios: Comenzar el desarrollo de los microservicios según la arquitectura definida en la fase anterior. Cada microservicio se desarrollará en paralelo.
- Integración Continua: Implementar un sistema de integración continua para garantizar que los cambios se prueben y se integren de manera constante.
- Seguridad: Implementar medidas de seguridad, como autenticación de usuarios, protección contra ataques y cifrado de datos.
- Pruebas Unitarias y de Integración: Realizar pruebas unitarias y de integración de manera continua para identificar y solucionar problemas a tiempo.

c. Pruebas y Ajustes: (4 semanas)

- Pruebas de Usuario: realizar la entrega a QA, para probar la aplicación y recopilar comentarios para realizar mejoras.
- Ajustes y Correcciones: Realizar ajustes basados en documentación de las pruebas realizadas por QA entre las mas usadas y recomendadas son las de regresión.
- Preparación para el Lanzamiento: Preparar la infraestructura para el lanzamiento, incluyendo servidores, almacenamiento en la nube y monitoreo.

d. Lanzamiento de la aplicación: (3 semanas)

- Lanzamiento Controlado: Realizar un lanzamiento gradual para un grupo limitado de usuarios y evaluar el rendimiento del sistema en producción.
- Pruebas de carga y estrés: este servicio puede ser subcontratado para garantizar el buen funcionamiento y performance de la aplicación.
- Pruebas de Vulnerabilidad: revisar con detalle el código fuente para identificar las vulnerabilidades que pueda tener el software y corregirlas de inmediato.
- Release: Administrar correctamente el versionado de la aplicación.
- Monitoreo Continuo: Establecer un sistema de monitoreo en tiempo real para detectar problemas y garantizar la disponibilidad constante del servicio.

e. Post-Lanzamiento: (contrato comercial con el cliente)

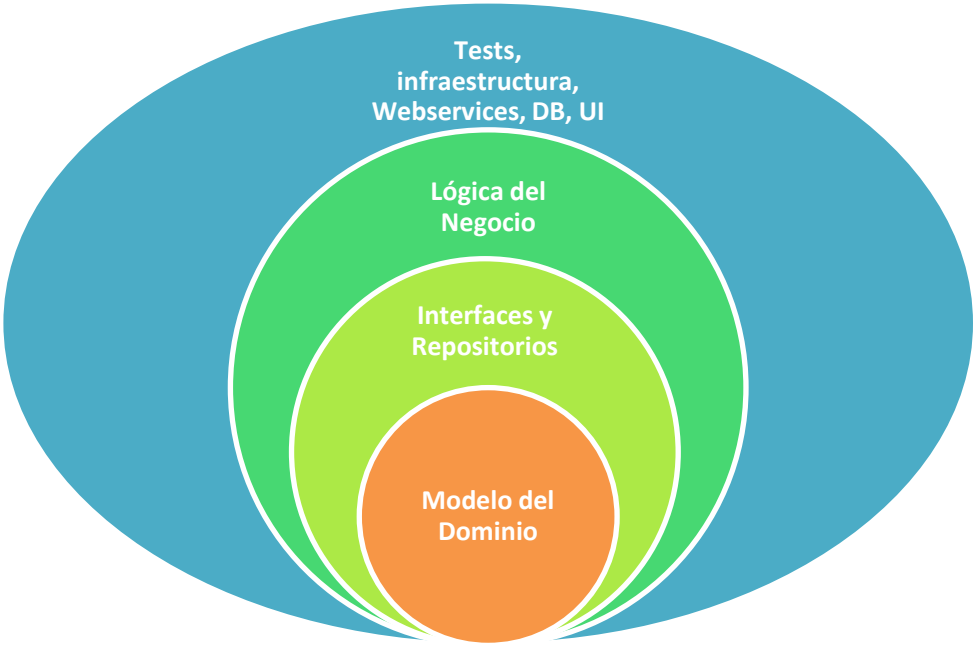
- Soporte y Mantenimiento: Proporcionar soporte técnico a los usuarios y abordar cualquier problema que surja.
- ANS: de ser necesario establecerlos en común acuerdo para garantizar el funcionamiento permanente de la app.
- Mejoras Continuas: Planificar actualizaciones periódicas para agregar nuevas características y mejorar la experiencia del usuario.

Las anteriores estimaciones se realizaron mediante un análisis PERT, basados en supuestos en los cuales tome como referencia mi experiencia para este tipo de

desarrollos, y lo más importante es contar con una célula experimentada para poder cumplir con la totalidad de los desarrollos.

2. Generar una propuesta técnica de orden arquitectónico no detallada de cómo resolver este problema y generar una estructura base (Diseño arquitectónico de alto nivel) que describa la solución. Casos de uso, características sistémicas, escenarios de calidad.

RTA/
Para atender dicho desarrollo, es importante definir una arquitectura limpia que permita el correcto funcionamiento de los desarrollos, la lógica del negocio, a la administración de los repositorios, Webservice, interfaces y test de la misma, por lo tanto, propongo una arquitectura "Onion Architecture", la cual cumple bajo mi criterio con el performance de la aplicación.



De acuerdo al gráfico anterior, podemos identificar que cada capa puede ser administrada sin problemas y con la mayor seguridad posible, para dicho desarrollo, teniendo en cuenta que requiere múltiples transacciones, multi usuario y pagos en línea integrando pasarelas de pago, mensajería y verificación de identidad permanente.

3. Describir los riesgos técnicos del proyecto.

   RTA/

   Dentro de la estimación de los riesgos del proyecto se identificaron los siguientes:

   - Estimación errada en los tiempos de desarrollo, es muy importante conocer el alcance del proyecto, el rendimiento de la célula de desarrollo, los proyectos compartidos entre la célula y demás puntos críticos para realizar una correcta planeación.
   - No realizar las pruebas de calidad adecuadas, aunque el área de calidad esté o no bajo la jerarquía del líder técnico, es muy importante definir los criterios de calidad que se aplicaran en la revisión a los desarrollos, esto impacta los criterios de aceptación de cada hito de entregables, el conocimiento del modelo de negocio para realizar las pruebas indicadas, la experiencia de los QA, el entendimiento de los agentes externos del proyecto tales como la integración de API'S, la interacción de servidores con la aplicación, versionado de los componentes de un Release.
   - Mockups incorrectos, la aceptación del encargado de producto debe evaluar los diseños de las interfaces de la app, para evitar reprocesos.
   - Integración de tecnologías desconocidas, sin una socialización previa con la célula, los arquitectos de software, el líder de producto y QA, en la cual se expongan todos los impactos técnicos que tienen, esto puede representar un riesgo muy alto y reprocesos en el desarrollo.
   - La falta de motivación de la célula, impacta considerablemente al proyecto, por lo tanto, se debe velar por mejorar las relaciones internas y atender cualquier eventualidad que pueda afectar el desempeño de cada miembro del equipo.
   - Involucrar a DEV nuevos en el equipo, la curva de aprendizaje puede ser muy tediosa y desviaría el rendimiento del equipo que estuvo desarrollando desde el comienzo.
   - Mal versionado de los Release, el control de versiones debe determinarse desde el comienzo del proyecto e implementarlo dentro de los repos, por lo tanto, si dichas acciones no se realizan a tiempo, se perderá el control de las versiones y, por ende, si se requiere de ser necesario un Roll Back, saber con exactitud el número de la versión y los componentes que se liberaron en dicho Release.

4. Describir cómo va a gestionar al equipo desde el punto de vista técnico, cómo va a medir y controlar el proceso y gobernar el alcance técnico.

RTA/

   - Definir las tecnologías que se van a usar para desarrollar la aplicación.
   - Gestión de permisos y accesos a las capas del proyecto para toda la célula.
   - Asignar un encargado para cada capa del desarrollo.
   - Identificar la especialidad de cada miembro de la célula, Back, Front, BD, etc.
   - Programar los Dailys para evaluar el rendimiento y apoyar a resolver los inconvenientes técnicos que se presenten.

- ❖ Verificación de buenas prácticas de desarrollo, para reducir brechas de seguridad y performance de la app.
- ❖ Verificación de librerías complementarias de desarrollos.
- ❖ Cumplimiento de código seguro, Ethical Hacking y código expuesto.

5. Establecer mecanismos y estrategias para lograr el objetivo de forma segura.

RTA/

- ❖ Tener claro el alcance, riesgos, costos y tiempos del proyecto.
- ❖ Asignar la cantidad de HU correctas a los miembros del equipo DEV, para evitar deudas técnicas.
- ❖ La estimación con la herramienta PERT será mi mayor aliado para una correcta planeación.
- ❖ Evitar sobre cargar a un recurso DEV.
- ❖ Asegurarse que cada miembro de la célula comprenda el modelo de negocio, y los criterios de aceptación de cada HU, con el fin de no desarrollar componentes con errores o incompletos.
- ❖ Comunicación asertiva y permanente entre todos los miembros del equipo.
- ❖ Medición (KPI'S) desempeño x recurso Vs tiempo de desarrollo y complejidad de las tareas asignadas.
- ❖ Reviews de cada entrega a producto y al cliente, para ir liberando componentes con base en la planeación definida.
- ❖ Retrospectivas dinámicas, para complementar el desempeño de cada entregable.

6. Describir cómo va a gestionar el proceso, prácticas, deuda técnica.

RTA/

- ❖ Definir qué se entiende como deuda técnica de Sprint.
- ❖ Hacer refinamiento y Feedback constante con el cliente.
- ❖ Priorizando los desarrollos pendientes en conjunto con la célula y el líder de producto, para crear los compromisos y ajustar los tiempos de las entregas.
- ❖ Proponer Sprints cortos que permitan cumplir con la totalidad de las HU referenciadas.
- ❖ Refinar el backlog para priorizar desarrollos, sin importar que se requieran crear varios Sprints para cumplir con los desarrollos.
- ❖ Identificar las épicas, desglosarlas, y desarrollarlas dentro de la estimación correcta de tiempos, recursos y dificultad.
- ❖ Tomando en cuenta que el área de calidad suele demorar sus pruebas debido a la revisión exhaustiva y tipos de pruebas aplicadas, los desarrollos se deben entregar con un periodo prudente para evitar demoras en las entregas validadas por calidad.

NOTAS:

- Documenta todos los supuestos que has establecido para el planteamiento de tu solución.
- Para resolver la prueba cuentas con 3 días a partir del momento en que recibes la prueba.

Recomendación: Organiza todo en capítulos y esmérate en documentar decisiones, diseño y procesos a usar. ¡¡¡La organización cuenta!!!

¡¡¡ Mucha suerte y gracias!!!