Análisis de la Plataforma Eventbrite

Ejercicio 2

Leonardo Araoz

Eventbrite

Como se cita en la página ¿Que es Eventbrite?

> Eventbrite es la manera más simple y confiable para vender entradas e inscribir participantes para cualquier tipo de evento.

Una plataforma de tal magnitud y ambicioso objetivo, con el alcance mundial que tiene, según mi entender debe lidiar temas relacionados con:

- Accesibilidad y Usabilidad
- Detección de Fraude y Seguridad
- Alta disponibilidad

Accesibilidad y Usabilidad

Con el afán de alcanzar a la mayor cantidad de personas de debe poner un gran esfuerzo en hacer esta plataforma de fácil acceso y uso para todos. Estrategias como:

- Seguir las guidelines de la Web Accesibility Initiative (WAI)
- Además de i18n (internacionalización) y L10n (Localización)
- Responsive Web Design además de la creación de Apps específicas para cada tipo de dispositivo
- En web, concentrarse en Web Performance Optimization (WPO) y en estrategias de SEO (Search Engine Optimization)

Detección de Fraude y Seguridad

Una de las características principales que afectan a la experiencia de usuario y a la confianza sobre la plataforma es la seguridad y la percepción de esta.

- El uso de https, utilizando los últimos estándares TLS y SSL.
- Adoptar y aplicar las normas de OWASP y CWE.
- Aplicar analítica en el comportamiento de los usuarios en la plataforma para modelar un perfil de usuarios potencialmente peligrosos.

Leonardo Araoz

Alta Disponibilidad

En eventos de gran magnitud o de gran expectativa, como por ejemplo Boca-River link, se necesita aplicar una ingeniería del software enfocada en la disponibilidad:

- El uso de CDN para el delivery óptimo de los assets, balanceadores de carga para una escalabilidad horizontal
- La implementación de micro-servicios para eludir un 'único punto de fallo'
- Optar por lenguajes de programación que proporcionen ventajas en ambientes distribuidos, concurrentes y paralelizables.
    - Ejemplos: Haskell, Scala y un enfoque funcional de Python
- Favorecer el uso de despliegues en arquitecturas Cloud y con contenedores (Docker) para así alcanzar un ciclo de Continuous Deployment.
- De la mano del diseño distribuido, se debe tratar incorporar al diseño tipos de base datos acordes a cada necesidad:
    - Con un enfoque en la velocidad => BD NoSQL (Ejemplo Cassandra)
    - Con un enfoque en la transaccionalidad => BD SQL (Ejemplo MySQL)

Adicionalmente:

- Aplicar una arquitectura centrada en el dominio de la aplicación y no tanto en los detalles de implementación, por ejemplo una Arquitectura Hexagonal (también nombrada: Clean architecture o Onion architecture)
- Utilizar un enfoque CQRS con el uso de un Command Bus

Sobre la Consistencia Eventual y aplicaciones de la IA

Combinando las dimensiones problemáticas de la Alta Disponibilidad y la Seguridad surge una temática en los sistemas altamente disponibles:

Consistencia Eventual

En el afán de brindar a los usuarios un entorno sin bloqueos ni tiempos de espera entre operaciones las soluciones distribuidas y las BD NoSQL demuestran sus virtudes.
Pero bajo el teorema CAP las bases de datos de tipo no relacional no destacan por proveer una alta consistencia. En este caso un enfoque mixto de uso de BD puede ser el equilibrio necesario en este caso.

Leonardo Araoz

Por otro lado combinando un poco las dimensiones de Seguridad, Accesibilidad y Usabilidad del producto, se podrían innovar con soluciones que hagan uso de la Inteligencia Artificial, el Machine Learning y el Big Data para crear modelos de aprendizaje que con el entrenamiento adecuado permitan crear:

- Bots de detección de comportamiento peligroso/fraudulento.
- Sistemas de recomendación de Eventos a usuarios de la plataforma
- Motor de sugerencias para los organizadores de Eventos basado en las acciones más efectivas de todos los eventos en el universo de información que tiene la plataforma.

Llegar a todos lados

Los eventos son sociales por naturaleza por lo tanto es una necesidad imperante llegar a la mayor cantidad de personas posibles.
Por lo que he podido relevar la plataforma actualmente nos permite integrarnos con Facebook, Twitter como redes principales.
Adicionalmente a las anteriormente nombradas existen otras redes sociales "de nicho" las cuales podrían tener un tratamiento especial acorde al tipo de evento por ejemplo:

- Youtube: los influencers audiovisuales regularmente ofrecen eventos.
    - La posibilidad de links especiales para suscriptores regulares o Patreons.
    - Assets para la creación del video promocional.
    - Networking con plataformas de publicidad en esta red social.
- Linkedin: Gran cantidad de eventos centrados en profesionales de distintas áreas corporativas se encuentran en EventBrite.
    - La posible integración con el sistema de certificaciones de LinkedIn es una gran oportunidad.
- Pinterest: El mundo del DIY, el arte y las manualidades está reflejado por esta influyente red social.
    - Posible integración de la acción "Inscribirse con un Pin".

Adicionalmente a las redes sociales nuevos modelos de interacción surgen con la llegada de nuevas tecnologías:

- Reserva de entradas con un asistente:

    - Google Assitant (Alexa, Siri, Google Home) permiten con interfaces de voz realizar cada vez más acciones, una de ella podría ser reservar un ticket por voz.
- Alertas de eventos Geoposicionales
    - Con el uso de GPS es posible crear alertas basadas en la ubicación sobre posibles eventos ocurriendo mientras nos movemos por la zona. (Al mejor estilo Pokemon Go)
- Uso de la Realidad Aumentada
    - Posiblemente con Spectacles o Google Glass Enterprise, o con el mismo smartphone para poder destacar a través de la pantalla donde se realiza el evento mostrando información relevante.
    - la misma tecnología también se pueden utilizar para los organizadores a elegir el mejor lugar para organizar el evento mostrando información importante de los posibles lugares dispuestos para eventos.

Herramientas para alcanzarlo

Sin duda para lograr todo este tipo de innovaciones es necesario utilizar herramientas adecuadas para el caso.
Desde mi perspectiva un ecosistema de desarrollo moderno debería poder cumplir con las reglas que propone Joel Spolsky link.

Pero a nivel general se debería optar por:

- Un Lenguaje de Programación con un gran ecosistema. Ej. Python
- Servidores de Integración Continua. Ej. Jenkins
- Sistemas de Control de Versiones. Ej. Git
- Testing y Documentación. Ej. `unittest` y Doxygen
- Guías de Estilo. Ej. link
- UML, en lo posible utilizado en una forma versionable. Ej. PlantUML