Universidad del Salvador

Facultad de Arte y Arquitectura

Licenciatura en Arte y Diseño Digital

Análisis del Buddy System en un Equipo de Diseño

Ética Profesional

Autor:
Matias Abarquez Mendoza

Profesor:
Alejandro Brianza

22/07/2022

Tabla de Contenido

Introducción

Una problemática que concierne a los nuevos profesionales a la hora de entrar a un equipo, es la adaptación al mismo, la desorientación que puede provocarle no saber a quién acudir, hacia dónde dirigirse en busca de respuestas. Si a esto le añadimos que debido a la pandemia del COVID-19 se empezó a implementar el trabajo remoto de forma permanente, la interacción social dejó de ser un hecho y un ingresante a la empresa tendrá estrictamente contacto virtual, tomando más relevancia la necesidad de un apoyo incondicional para la orientación. Es por eso que todos los equipos deberían implementar lo que se llama Buddy System.

Este sistema plantea el emparejamiento de una persona del equipo con el reciente ingresado. Para que lleve una mejor resolución frente a los problemas de ser nuevo, el buddy tiene que tener conocimiento del contexto general de la empresa. El líder del proyecto es quien tiene que ser el encargado de acomodar este emparejamiento para un resultado óptimo.

La implementación de esta modalidad en un equipo de diseño liderado por un líder de proyecto, facilita la inserción de nuevos profesionales de una manera efectiva, eficiente y de mejor respuesta frente a los problemas reales que tiene que afrontar una persona en una nueva empresa. Así mismo, quien hace de buddy consigue una evolución personal en diversos aspectos.

En muchas empresas, generalmente de tamaño considerable, suelen usar esta metodología del Buddy System ya que hay mucho por aprender y tienen la necesidad de que cada nuevo profesional se adapte lo más rápidamente posible. Sin embargo, en el presente trabajo se analizará una empresa en particular que se llama MercadoLibre.

Esta compañía se fundó en Argentina a principios del año 2000. Nació como una empresa de venta y compra de artículos, aunque actualmente tiene otros servicios, debido al avance tecnológico, como recarga de celulares, recarga de tarjeta de transporte, transferencias monetarias, entre otros. A medida que avanza la tecnología se van agregando nuevos, dividiendo internamente las unidades de la empresa. Actualmente se

encuentra en países económicamente fuertes como Argentina, Brasil, Uruguay, México, Chile. Si bien los mencionados anteriormente son los que pisan con fuerza, MercadoLibre sigue expandiéndose.

Resumiendo, se tomará MercadoLibre como estudio de caso a analizar. Ubicándonos en tiempo, la presente tesis se centrará desde la existencia de la pandemia de COVID-19 que se expandió al mundo en el año 2020 hasta el año actual, 2022, ya que la pandemia aún no ha terminado. Por último, y no menos importante, nos ubicamos en Latinoamérica como lugar geográfico.

Mi interés por el tema nace a partir de mis dos experiencias en las cuales tuve el proceso de inducción y a su vez me asignaron un buddy –compañero en español– sin entender cuál era su finalidad.

En el primer caso en el cual se me asignó un buddy, recién arrancaba la carrera de programador y entraba a una empresa multinacional, sin saber toda la burocracia que tiene por detrás –diferente a mi primer trabajo como pasante en una Embajada–.

El rol que ocupaba mi buddy era diferente al mío, sin embargo, era del mismo equipo. Ella se especializaba en atención al público y capacitar a los clientes de la aplicación web que desarrollábamos y yo me encargaba del área de desarrollo. A su vez un compañero –hoy en día amigo–, también ingresante, le asignaron un buddy que era desarrollador. Yo veía como su buddy, quien no era del equipo, lo ayudaba a configurar toda la computadora para poder trabajar mientras que mi buddy, que sí era de mi equipo, ni me dirigía la palabra. Debido a lo anterior, le pedía ayuda a mi compañero de trabajo quién tenía el mismo rol que el mío. En particular puedo decir que fue una experiencia negativa y hasta aquí creía que el término buddy era simplemente una persona impuesta por alguien porque así se manejaba la empresa.

Hablando de mi segunda experiencia a través de un buddy, fue en la empresa en la que me encuentro actualmente –siendo el año 2022–. Al haber tenido una pésima experiencia, pensé que iba a tener los mismos resultados. Una persona impuesta por alguien para que me ayude pero que no tenga tiempo de hacerlo o no quiera hacerlo por diversas cuestiones.

Puedo decir que la empresa tuvo otros valores, el equipo se comportó diferente desde el principio, una tranquilidad y un positivismo que no había encontrado en los 7 meses en que estuve en aquel equipo donde tuve un buddy que no cumplía la función como tal.

En contraste, aquí supe posteriormente que quién decide quién va a ser el buddy del nuevo profesional ingresante es elegido por el líder de proyecto del equipo.

En el primer caso, había preguntado cómo se había elegido el tema del buddy, un amigo me había comentado que se debía a habilidades blandas. Básicamente, relaciones personales –lo cual no me convencía tampoco porque no me hablaba–.

Puedo decir que fue una experiencia totalmente positiva ya que mi buddy era de mi propio rol, tuve meses en el que el aprendizaje fue en continuo crecimiento, tanto a nivel conocimiento de la empresa como también a nivel estructural en la organización de los proyectos. No quiero extenderme y explicar cuestiones técnicas en las cuales me ayudó en mi recorrido desde el día uno y se preocupó de poder dedicarme tiempo. Culminando esta experiencia positiva, está demás decir que pude entender a mi compañero –de la primera experiencia– quien tuvo a un buddy del mismo rol a pesar de ser de diferente equipo.

El onboarding –también llamado inducción–, es el proceso por el cual un nuevo profesional se adapta a su nuevo espacio de trabajo, reconoce las formas y las herramientas en las que trabaja su nuevo equipo, así como las normas generales de la empresa. A su vez, tiene que haber una persona capacitada que ayude en esta adaptación.

El proceso de onboarding perdura por un breve período de tiempo, mínimamente un mes, para que la persona pueda desenvolverse con facilidad en su entorno laboral.

Laurie Mullins (2005) afirmó que el programa de inducción debería ser preparado para ayudar a los nuevos miembros del equipo a familiarizarse con el nuevo ambiente, prepararlos en su nuevo puesto de trabajo y establecer buenas relaciones interpersonales con sus pares. La inducción

debería ser más allá de un contrato psicológico en pleno proceso de reclutamiento y selección.

Al mismo tiempo dice que es importante recordar que la gente no absorbe mucha información de una vez, particularmente puede ser una situación rara e incómoda. El programa de inducción debería ser planeado cuidadosamente e implementado en un tiempo razonable. Implicará la cooperación de managers, supervisores y colegas. Dentro de un programa de inducción efectivo son recursos de ayuda la información sobre inducción, presentaciones y un mentor que actúe como buddy, para guiar y ayudar al nuevo miembro del equipo (p.817).

Lo explicado anteriormente por el autor, es parte del proceso de reclutamiento y selección del candidato. Si bien es una excelente definición de cómo sería el proceso y las particularidades que el buddy debería enfrentar, me encuentro en posición de alejarme de la postura de considerar al buddy como un mentor.

¿A qué se debe que no consideraría a un buddy como un mentor? Porque son dos términos que, si bien pueden confundirse, cada uno tiene un objetivo diferente.

Un mentor –también llamado tutor en español– es una persona más experimentada, confiable, con más conocimiento que guía a otra de menos experiencia y menos conocimiento. La guía y la dirección a la que apuntan es en un aspecto que el mentor tuvo éxito. Así, el mentor se encuentra calificado para replicar el mismo o más éxito del que tuvo a otra persona. A la persona que el mentor acepta para guiar y dirigir se le denomina mentee.

La relación entre mentor y mentee, como también al proceso de guiar y dirigir se denomina mentoría –sólo en español, en inglés tienen dos palabras diferentes. También suele escucharse como tutoría– (Osiri, 2020).

Cada equipo busca la mejor manera de integrar a un nuevo empleado a su forma de trabajo, estableciendo las pautas necesarias para que lleve a cabo sus responsabilidades. Sin embargo, todo empleado puede estar

desorientado ya que no tiene a quien recurrir permanentemente y cada pregunta puede ser de diverso contenido y tendrá que ir de persona en persona hasta dar con la correcta. Por este mismo motivo es que se creó el Buddy System, no sólo para orientar al nuevo integrante del equipo sino también para prevenir futuros errores y saber a quién acudir. Seguramente habrá alguien que decida quién es la pareja del nuevo profesional. Se pueden pensar diferentes variables a la hora de elegir al buddy. Puede que quien decida lo haga por sorteo entre los miembros, que la persona elegida sea porque tiene habilidades para desarrollarse socialmente, o alguien que necesite desarrollar sus habilidades comunicacionales, también hay gente que lo hace de manera voluntaria.

Hay una persona que es la más capacitada para realizar este trabajo y debe hacerlo como una responsabilidad crítica, el Project Leader [PL]. Esta persona tiene la misión de liderar el proyecto en sí, saber sobre las cualidades de cada uno y qué habilidades mejorar. Por lo que cumple un factor determinante a la hora de elegir un buddy por ser quien evalúa a cada integrante del equipo. Es aquí donde yace la siguiente pregunta:

¿De qué manera implementar el Buddy System de una manera eficiente y eficaz?

Para implementar el Buddy System de una manera eficiente y eficaz, en la cual una persona es emparejada con otra, el buddy requiere que tenga el mismo rol y esté en el mismo equipo. Esto facilita la capacitación, puntualmente en las responsabilidades que tiene el ingresante en el día a día y no está únicamente limitado a las generalidades de la empresa.

El presente informe aporta en el ámbito profesional una correcta manera de realizar el onboarding de un nuevo integrante del equipo a través del Buddy System. Se toma en consideración las variables con las que contará el buddy que posteriormente será designado como tal al nuevo profesional. Estas consideraciones incluyen desde las habilidades blandas –también llamadas soft skills en inglés– para poder socializar, hasta habilidades técnicas que el rol del nuevo ingresante del equipo requiera,

para así poder llevar a cabo un ida y vuelta sobre cómo está organizado la estructura de los proyectos internos –frameworks, librerías, IDE–. El rol del buddy juega un papel muy importante a la hora de hacer transferencia de conocimientos hacia el nuevo profesional.

Por otro lado, en el ámbito de lo social, se aporta lo que un nuevo profesional del área de diseño tiende a esperar en el día a día de su buddy –en caso que la empresa implemente este sistema– con respecto a las tareas y los conocimientos necesarios para adaptarse a su nuevo equipo, añadiendo que hoy en día el trabajo remoto es una realidad dentro de una organización burocrática y ya no solo los freelancers trabajan de esa manera. El ingresante conocerá diariamente cómo es el mecanismo dentro del ecosistema empresarial, las pautas que hay que tener en cuenta, las consideraciones –qué se puede o no hacer con la herramienta de trabajo que es la computadora–, cómo se relacionan entre sí los diversos equipos que puedan coexistir, el flujo del equipo designado.

Objetivos

Teóricos

General. Emparejar un integrante del equipo y el nuevo integrante con el mismo rol.

Específico.

1. Analizar el rol del nuevo integrante del equipo.
2. Analizar los integrantes del equipo que tengan el mismo rol que el nuevo profesional.
3. Determinar a un integrante del equipo para capacitar al nuevo profesional que ingresa.

Prácticos

General. Diseñar un video audiovisual sobre el beneficio de implementar el Buddy System de una manera eficiente, eficaz y concisa.

Específico.

1. Demostrar una solución fallida sin incluir un emparejamiento de rol ni equipo.
2. Demostrar una solución parcial sin incluir emparejamiento de rol.
3. Demostrar una solución parcial sin incluir emparejamiento de equipo.
4. Demostrar la solución definitiva y efectiva incluyendo emparejamiento por rol y equipo.

Desarrollo

> Actualmente, la empresa afirma que su ecosistema:
> Está integrado por dos grandes unidades de negocio: el ecommerce y las fintech. Dentro de estos dos grandes mundos, Mercado Libre y Mercado Pago, se agrupan distintas empresas orientadas a mejorar la experiencia del usuario: Mercado Envíos, Créditos, Mercado Shops, Ads y VIS (MercadoLibre, 2022).

Mercado Libre es el marketplace tal como lo conocemos todos. Es el servicio de compra y venta de artículos que opera a nivel latinoamérica.

Por otro lado, la unidad de Mercado Envíos es la encargada de "brindar soluciones de logística para mejorar la experiencia a millones de vendedores y compradores" (MercadoLibre, 2022). Aquí se puede hacer una distinción de los transportes de mercadería, recientemente en algunos países se está empezando a utilizar flota de transporte eléctrico a favor del cuidado ambiental.

Si hablamos de Mercado Libre Vehículos, Inmuebles, Servicios [VIS], estamos haciendo mención a la unidad que están conectados al marketplace –Mercado Libre– "pero tienen un modelo de negocio diferente al resto de la plataforma: son categorías no transaccionales, es decir, que la compra-venta se completa fuera de la plataforma." (MercadoLibre, 2022).

Por el lado de la fintech[1], Mercado Pago, se puede decir que "brinda la cuenta digital más completa para aquellas empresas, emprendimientos o personas físicas que desean gestionar su dinero de manera segura, simple y cómoda con una gran variedad de posibilidades" (MercadoLibre, 2022) en

[1] "Es un sector integrado por empresas que utilizan la tecnología para mejorar o automatizar los servicios y procesos financieros". Fuente: Maestre, Raúl Jaime (11 de marzo de 2022). *Qué es fintech y por qué es el futuro de las finanzas*. iebschool. Recuperado el 04 de julio de 2022 de https://www.iebschool.com/blog/que-es-fintech-finanzas/

los países que se encuentran habilitados, ya que en no todos es posible usar todos los servicios que se brindan por parte de la empresa. Un ejemplo concreto es la reciente utilización de criptomonedas, actualmente sólo se encuentra en Brasil.

En cuanto a Mercado Crédito, es una plataforma crediticia que "otorga préstamos y soluciones financieras a usuarios de los productos de la compañía." (MercadoLibre, 2022). Se basa en un score –puntaje– propio del historial de ventas y compras dentro de la unidad de Mercado Pago para poder evaluar si el usuario está apto para conseguir un préstamo.

La unidad de Mercado Shops permite “crear una tienda online propia de manera simple y ágil, con todos los beneficios del ecosistema de Mercado Libre.” (MercadoLibre, 2022). Esto implica facilitar que aquella persona, empresa PyMEs que quieran utilizar una tienda online del estilo de MercadoLibre puedan crearla para su propio beneficio.

Por último, se encuentra Mercado Ads, esta unidad se encarga del desarrollo de “soluciones de publicidad dentro del e-commerce líder en América Latina. Permite a vendedores y marcas aprovechar el alto volumen de visitas que tiene Mercado Libre para enviar tráfico” (MercadoLibre, 2022).

Implementar un Buddy System en el lugar de trabajo es sencillo. Algunos de los pasos son los siguientes.

Decidir una estructura. Escribir el propósito del programa y el objetivo para la empresa y los empleados. Anotar el período de tiempo de cada emparejamiento y cualquier regla que aplique en las relaciones con el buddy.

Establecer las expectativas. Establecer tareas específicas y expectativas en el Buddy System. Definiendo las expectativas tan claras como sea posible, se puede seleccionar a la gente correcta para el programa y seguir con el proceso.

Encontrar a los participantes. Encontrar voluntarios para asistir en el reclutado de nuevos empleados. Es importante tener voluntarios de múltiples

departamentos porque limitarlo al sistema de recursos humanos no es lo ideal. El Buddy System es más efectivo cuando los buddies tienen un propósito general y pueden compartir sus tareas diarias con el nuevo empleado. En caso de no encontrar voluntarios, encontrar la forma de incentivar el programa.

Emparejar al nuevo empleado. Preguntar al nuevo profesional algunos temas para que ayuden a emparejarlos con otros que tienen objetivos y personalidades similares (Indeed, s.f.).

Llegado a este punto, algunos autores indican como siguiente paso el hecho de crear una checklist –una lista de items que vas tildando a menudo que se completen– para llevar el día a día con el nuevo profesional que te asignaron. Sin embargo, no estoy de acuerdo en seguir una pauta estructurada, aunque puede ser de ayuda. Basada en mi experiencia, el hecho de tener una checklist es opcional y puede ser de ayuda, aunque no es imprescindible. Es por ello la razón de que no se agregó como un paso, pero se hace mención para quien lo necesite.

Las siguientes son las características a tener en cuenta para que una persona pueda ser elegida como buddy al utilizar el Buddy System.

Destacada performance de trabajo. El buddy tiene que ser un buen ejemplo y haberse destacado en la performance de su trabajo en el pasado y en el presente.

Habilidades en la nueva posición. El buddy tuvo que tener la posición en el pasado o realizar similares responsabilidades.

Tener buenas habilidades de comunicación. Es importante que el nuevo empleado sea emparejado con alguien que sepa comunicarse efectivamente y pueda escuchar activamente entendiendo lo que el nuevo profesional necesita.

No ser un manager. El propósito del Buddy System es emparejar personas con quienes tengan similares roles, así como un supervisor no sería un buen buddy.

Sea accesible. Es importante que el buddy tenga tiempo disponible para asistir a alguien mientras hace sus responsabilidades.

Voluntad para participar. Está bien que la gente rechace la oportunidad de ser un buddy. Si están forzados a participar, el nuevo empleado es visto como una carga al buddy, y esto puede tener consecuencias indeseadas.

Los buddies no significa que sean supervisores o mentores.

Algunos puntos a destacar en el programa son los siguientes.

Los buddies se suponen que no son expertos. Está bien que ellos no sepan una respuesta y necesiten redirigir la pregunta a alguien más.

Las relaciones no son inmediatas. Algunos nuevos empleados no desarrollarán relaciones inmediatamente. Hay que ser pacientes.

Tomarlo con calma. No abrumar a los nuevos empleados con información y procesos. Hacerlo de a poco, en una transición continua en sus roles.

Tener tiempo para escuchar. Escuchar a los nuevos empleados y aplicar lo que dicen para ajustar lo necesario en el futuro. Esto incluye aprender a cómo comunicarse efectivamente y descubrir qué motiva al nuevo profesional.

Registrar nuevas ideas. Parte del programa es aprender de lo que brinden los nuevos empleados, así el buddy debería tomar nota acerca de los tips e ideas para mejorar la empresa.

Mantenerse positivo. Los buddies no deberían criticar a sus supervisores o la cultura de la compañía. Haciendo eso pueden hacer que el nuevo empleado se sienta incómodo y se pregunte si aceptaron el trabajo correcto (Indeed, s.f.).

El Buddy System beneficia el entorno de trabajo de diferentes maneras. Cada empleador lo implementa con diferentes objetivos. Algunas de las maneras que ayuda el Buddy System a la compañía pueden ser las siguientes.

Bienvenida a nuevos empleados. La primera semana en un nuevo lugar de trabajo es de puros nervios. Los nuevos profesionales contratados

están en un lugar desconocido y no saben nada de sus compañeros. Todavía están aprendiendo que se espera de ellos y pueden cometer algunos errores mientras se adecúan a su nueva posición.

Emparejando a los nuevos empleados con aquellos que ya llevan un tiempo en la empresa puede ayudarlos a sentirse más aliviados, y pueden aprender los procesos y cómo funciona la empresa más rápidamente. El Buddy System puede ayudarlos a asimilar la cultura de la compañía dejando que pregunten y construyendo relaciones interpersonales. Los nuevos contratados que reciben un buddy sienten que son parte de la comunidad más rápido.

Incrementar la confianza del empleado. Sea que un nuevo empleado tenga experiencia en su posición o no, cada cambio de trabajo requiere una curva de aprendizaje. Tu empresa no hará nada de la misma manera a lo que el nuevo profesional está acostumbrado, de esta forma el buddy puede aplicar su experiencia pasada como nuevo empleado y las habilidades. Esto construye confianza más rápido que dejar al empleado a su merced y que descubra las cosas por su cuenta.

El feedback constructivo es mejor recibido si viene de un buddy que de un superior. Los nuevos empleados a menudo temen que, si preguntan demasiado o reciben muchas críticas de cargos superiores, se sientan que no encajan en la empresa. La crítica es menos formal si viene de un buddy, y los nuevos empleados pueden relacionarse a las anécdotas que el buddy comparte acerca de cómo encontraron éxito en sus roles.

Incremento de productividad. Los nuevos empleados son más lentos en sus puestos porque todavía están aprendiendo. Es normal esperar que los nuevos contratados no sean tan productivos como los que ya tienen experiencia en la empresa. Tener un buddy impulsa a los nuevos a mejorar la curva de aprendizaje e incrementar la productividad.

Si se asegura que los empleados tengan el apoyo necesario utilizando el Buddy System, estarán más felices, con más confianza en su trabajo y más productivos. Además, estarán más dispuestos a compartir sus ideas ya que quizá no estarán cómodos compartiéndolas con sus jefes debido a la timidez. Las fuertes relaciones de trabajo ayudan a la

organización a tener una mejor colaboración y a una comunicación más efectiva.

Mejora de retención de empleados. El costo de adquirir nuevos empleados a menudo es más costoso que el hecho de retener a uno. Los empresarios intentan retener empleados ofreciendo salarios competitivos, paquetes de beneficios, oportunidades de carrera y entrenamiento. El Buddy System es una manera de proveer entrenamiento mientras incorporan a los empleados a la cultura corporativa. Cuando la gente se sienta cómoda con sus compañeros de trabajo, tienden menos a dejar la empresa.

Ganar visión e innovación. El intercambio mutuo de ideas puede crear oportunidades de innovación y creatividad. Los manager acceden a nuevos tips, procesos y procedimientos que antes no consideraban (Indeed, s.f.).

Como afirma Choudhary de Zavvy, la principal diferencia entre mentor y buddy radica en que el buddy es quien te acompaña en los primeros pasos en tu llegada al equipo, mientras que los mentores son personas experimentadas en el mismo puesto quien te ayudará a corto, mediano y largo plazo en tu carrera profesional (Choudhary, 2021). Es decir, se basa en mejorar tus habilidades técnicas para que a futuro progreses con respecto a tu seniority.

Ahora, en relación al Buddy System, el PL tiene que evaluar en qué puesto –rol– se va a desempeñar el nuevo profesional que formará parte del equipo. Ya que así podrá asignar un buddy que tenga el mismo rol y puedan compartir conocimientos. Si es alguien que ya tiene sus años dentro del equipo –ni hablar dentro de la empresa–, puede ser un mentor para la carrera profesional al mismo tiempo que es buddy.

Para hacer el correspondiente análisis de porqué es importante un Buddy System se necesitan aclarar los aspectos en la estructura de un equipo de trabajo correspondiente a una aplicación web.

Para tener un buen equipo de desarrollo, COR (s.f.) dice que hay que considerar los siguientes factores:

> Cubrir todos los roles y habilidades. [...] Cada rol tiene sus propias responsabilidades y habilidades específicas esenciales para

que un proyecto tenga éxito. Sin cubrir todos esos roles, el equipo se arriesga a sufrir retrasos, a la baja calidad y a los clientes insatisfechos.

Comunica los objetivos y los indicadores claves de rendimiento –o KPIs, por sus siglas en inglés–. [...] Los objetivos deberían ser específicos, medibles, realizables, realistas y cronometrados. Establecer KPIs para el equipo y para los individuos también debería ser una forma eficaz de buscar el mejor rendimiento en tu equipo.

Contrata un talento diverso. […] Al buscar talento de distintos orígenes y formaciones, ayudarás a conseguir diferentes ideas, perspectivas y valores para el equipo. […]

Haz que la información esté disponible. Para que entregues los proyectos a tiempo, debes asegurarte de que los miembros de tu equipo tengan todas las herramientas necesarias para acceder a la información. Hacer que la información esté disponible implica una mayor transparencia en el progreso de tu equipo, y les permite a los miembros ver dónde están habiendo avances y dónde habría que mejorar las cosas. […]

Automatiza procesos. La automatización significa que los miembros del equipo tendrán más tiempo para invertir en actividades facturables, y eso implica avances más rápidos en los proyectos para poder trabajar en otros. […] Deberías darle a tu equipo las herramientas para automatizar todas esas tareas administrativas que consumen tanto tiempo. Esto también te ayudará a cuidar del bienestar de los miembros de tu equipo, ya que liberarás su tiempo.

Un proyecto tendrá diferentes roles en el equipo de desarrollo. Suponiendo que fuese de una aplicación web y que sea la mínima cantidad de integrantes, debería tener un frontend developer, un backend developer, un PL o PM y un TL. Hay ocasiones en que el frontend developer y el backend developer es la misma persona, llamado FullStack Developer, aunque actualmente se encuentran separados por tema de tiempo, de preparación.

El diseñador UX tiene como objetivo visualizar al “usuario final interactuando con el producto, y hacen que este sea fácil de utilizar y se enfocan en todos los aspectos de la experiencia: usabilidad, funcionalidad y rendimiento.” (COR, s.f.).

El diseñador UI “se enfoca principalmente en el software y en cómo se ve y se siente para el usuario. Necesita que sea intuitivo y directo.” (COR, s.f.).

Ambos diseñadores se complementan “ya que son los que defienden las necesidades del usuario”. Además, deben contar “con un estilo creativo y un fuerte enfoque en el diseño. Deben ser analíticos y pensar con originalidad.” (COR, s.f.). Muchas veces estas dos especialidades se concentran en una sola persona. Pueden formar parte del equipo propio de IT, o pueden tener su propio equipo de diseño, aunque tendrán que interactuar con el equipo de desarrollo de software para recibir feedback y entender qué se puede y qué no aplicar.

Se puede definir como Project Leader –o Líder de Proyecto en español– [PL] un profesional que lidera y se asegura que el proyecto progrese. El PL se compromete con el equipo, lo motiva, y se encarga de lo que necesiten y de mantener un ambiente de trabajo relajado y productivo (Indeed Editorial Team, 2020). Este rol es uno de los más importantes dentro del equipo, ya que se encarga de ser la cara visible frente a cualquier problema que surja dentro del equipo y se encarga de tomar las decisiones con respecto a los temas más importantes (Cole, Roby & Barker, Stephen, 2009).

El Project Manager [PM] –manager del proyecto– tiene una sutil diferencia con respecto al PL. El rol de PM se encarga de llevar a cabo las tareas del proyecto de manera ordenada, se centra en el proyecto en sí mientras que el PL se encarga de la parte humana de los miembros del equipo. Es quien coordina el proyecto con las reuniones, las tareas, y lo esencial para que un proyecto tenga éxito. No quiere decir que un PL no pueda hacer las tareas del PM, pero en este último caso está más involucrado con la parte del proyecto.

Según la experiencia de Boujon (2019) cada organización entiende el rol de Technical Leader [TL] –Líder Técnico– “dependiendo de diversos factores, como puede ser la madurez de la organización, las necesidades puntuales de la misma, la dinámica de trabajo y la cultura organizacional”. En líneas generales define a un TL como aquél que “debe tener buenos skills técnicos, debe contar con experiencia en el desarrollo de software, pero no necesariamente debe ser el mejor desarrollador.”. Adicionalmente agrega que cualquier líder debe contar con habilidades blandas –soft skills– .

Entre los soft skills más destacados, Boujon (2019) explica que el TL tiene que comprender “lo que le sucede al equipo y brindarle ayuda o simplemente prestar el oído, genera empatía y hace que la dinámica de trabajo en el día a día simplemente fluya”.

Además de empatía, el TL tiene que generar motivación a los integrantes del equipo, al respecto Boujon (2019) explica que “darles la libertad y el espacio a la hora de implementar soluciones, hará que ellos se sientan comprometidos y valorados”.

Un punto crucial es el hecho de la comunicación, el mismo autor afirma que tiene que hacer la difícil tarea de explicarle a los desarrolladores, y que entiendan, cómo va el proyecto, en qué punto están y cuáles son las expectativas del cliente. Más aún, tiene que tener la capacidad de poder comunicarse con el cliente, ser el receptor de cada mensaje y poder transmitirlo a quien corresponda (Boujon, 2019).

Otra característica destacable que tiene que tener un buen TL, Boujon (2019) afirma que debe poseer visión holística, esto hace referencia a “no solo conocer el software que se está desarrollando, sino también entender en qué parte del negocio encaja, quienes son sus usuarios y los stakeholders en general y los objetivos tanto del software como del proyecto y hacer el esfuerzo por entender cómo todo esto se alinea” para tener “información a la hora de tomar decisiones, identificar los riesgos, tratar de allanar el camino lo máximo posible para evitar trabas y pérdidas de tiempo”. Esto de tomar decisiones tiene que ver con evaluar la escalabilidad del

proyecto, la optimización de recursos y cuál sería la mejor manera de agregar nuevas funcionalidades al proyecto.

Otro punto fundamental tiene que ver con crear conocimiento colectivo. “Es muy común que haya gente que se especialice en una sola cosa o que sólo quién desarrolló una funcionalidad específica sabe qué hizo, cómo lo hizo, cómo funciona y por qué. El problema aquí es ¿Que sucede si esa persona no está y hay que incorporar cambios?” (Boujon, 2019), para solucionar esto el TL debe diversificar el conocimiento de los proyectos por todos los integrantes del equipo.

El último aspecto tiene que ver con la capacidad de gestión. La tarea del TL será agendar y participar de reuniones con el cliente para definir objetivos y/o requerimientos. Otros casos definir tareas más específicas, estimar tiempo, planificar sprints” (Boujon, 2019). Con respecto a lo último, basado en mi experiencia tengo que discernir un poco. La planificación de las tareas para un sprint –se hablará más adelante sobre la metodología scrum que involucra este concepto– las hace todo el equipo, aunque tenga mucha influencia el TL.

En resumen, el TL “es el nexo entre el negocio y las cuestiones técnicas, entre los tomadores de decisiones y los desarrolladores” (Boujon, 2019).

Las responsabilidades de un frontend se pueden agrupar en las siguientes.

El diseño, la creación y el mantenimiento de sitios web. Hay muchas herramientas para realizar el diseño de un sitio web, incluyendo papel y lápiz. Incluso es posible tener un diseñador que te provea el diseño. Sin importar de donde provenga el diseño, para implementarlo se va a tener que usar HTML, CSS y JavaScript (Myers, 2020).

Mantenerse al tanto de las novedades. [...] Con adaptaciones de estándares, con nuevas tecnologías, enfoques, técnicas apareciendo todo el tiempo y al mismo tiempo se eliminan otros aspectos, el ambiente cambia continuamente. Estar actualizado con estas nuevas herramientas permite mejorar futuros trabajos y mantener los anteriores (Myers, 2020).

Comunicación. Es importante que los desarrolladores estén al tanto de los requerimientos tanto de clientes como de potenciales usuarios. Posiblemente también se comunique uno mismo a futuro, por lo que la documentación y el código legible ayudará en este aspecto (Myers, 2020). Cabe aclarar que esta habilidad tiene que ver con todos los desarrolladores, sin importar la especialidad que poseen, es decir esta es una cualidad importante tanto para el frontend como el backend.

Testeo y validaciones. [...] Cualquiera sea la manera de testear, tener un plan es el primer tema a definir. Validaciones del lenguaje de marcado puede ser el primer punto importante, será la primera responsabilidad el producir lenguaje de marcado válido, ya que no todos los browsers –navegadores– lo tratan de la misma manera (Myers, 2020).

Un desarrollador backend es quien programa en cualquier lenguaje de programación para construir lógica de negocio funcional y útil para aplicaciones web. Adicionalmente, los desarrolladores backend realizan la conexión entre el frontend y la base de datos (Ramotion, 2021).

Las siguientes son las principales responsabilidades que tiene el desarrollador backend.

Proveer información a los usuarios. A pesar del diseño de la aplicación, los consumidores constantemente solicitan información mientras usan la aplicación (Ramotion, 2021).

Combinar y transformar la información. Cualquier información que necesite la aplicación puede venir de cualquier fuente, generalmente referidos como base de datos. En este sentido, el trabajo del desarrollador backend es localizar información puntual de lo que el usuario está buscando dentro de todas las bases de datos y solo combinar lo que necesita para producir la consulta solicitada (Ramotion, 2021).

Una aplicación web es capaz de localizar información exacta requerida por el usuario. Además, el esqueleto de la tecnología backend es constantemente optimizable y los desarrolladores a menudo agregan nueva información o información que es solicitada por los usuarios (Ramotion, 2021).

Finalmente, después que el backend recolectó y combinó toda la información solicitada, debe enviarla al usuario. Sin embargo, el backend necesita “traductores” para convertir el código puro en lenguaje humano. Para hacer eso, las APIs y el frontend entran en juego (Ramotion, 2021).

La diferencia entre el desarrollador frontend y el backend radica en que el desarrollador frontend es el miembro del equipo que más impacto tiene frente al usuario, a pesar de que no se encuentran tan a menudo. Mientras que el desarrollador backend se centra en la lógica de la aplicación y usará diversas herramientas para mantener esa lógica durante la interacción con el usuario, el desarrollador frontend utiliza un set diferente para asegurar que la lógica se presente a los usuarios de manera que tenga sentido y con estética satisfactoria (Myers, 2020).

A la hora de empezar un proyecto de cero hay que considerar que cada uno es particular para solucionar un problema y no es la misma tecnología aplicada. De esta manera, el stack tecnológico va a ser dinámico. A su vez, es necesario tener en consideración diferentes conceptos. Entre ellos están los que Aulab menciona.

Escalabilidad. La herramienta debe ser usable de la misma manera tanto cuando el proyecto es pequeño como cuando va a crecer considerablemente.

Seguridad. Cuando se comparten herramientas de trabajo y datos sensibles, el equipo y la empresa tienen que saber trabajar siempre con absoluta seguridad.

Facilidad de uso. La herramienta tiene que simplificar el trabajo y no complicarlo.

Portabilidad. El programador Web a menudo trabaja desde casa o en la oficina, la herramienta tiene que permitir poder trabajar dinámicamente desde cualquier sitio, garantizando los mismos estándares de uso.

Funcionalidad. Las herramientas utilizadas deben aportar un incremento de velocidad.

Codementor explica que el pair programming –programación en pareja– es la práctica de trabajar emparejados en determinadas tareas. Añade que usualmente, uno se imagina dos desarrolladores en una misma computadora, compartiendo teclado, pero debido a la popularidad de trabajar remoto, ahora es posible hacer pair programming desde miles de kilómetros alejados.

Al utilizar esta práctica, Codementor dice que ambos desarrolladores emparejados deben compartir pensamientos para que la técnica resulte efectiva. El éxito radica en la comunicación, así como también en las habilidades de programar, dos cabezas piensan mejor que una.

Al utilizar esta técnica, Codementor hace mención de los siguientes beneficios.

Producir mejores soluciones. Implica que es posible detectar problemas más rápido e identificar potenciales bugs por ambas personas en vez de una. Además, explica que ambas personas discuten y evalúan acerca de la solución a implementar.

Por otro lado, Codementor menciona otro beneficio, el hecho de compartir conocimiento y contexto durante el emparejamiento. Esto quiere decir, que a la vez que programan es posible entender el contexto del proyecto por parte de más de una persona del equipo. Debido a lo anteriormente mencionado, si acaso una persona deja el equipo por la razón que fuese, hay otro desarrollador que entiende el código. Sin esta técnica, haría falta tiempo extra y agendar reuniones para poder transmitir conocimientos sobre lo codeado.

Por último, Codementor menciona el aprendizaje y el desarrollo de habilidades mutuamente. El beneficio más importante es el hecho de aprender del otro. Si un desarrollador tiene más seniority que el otro, es la mejor manera que el de menor seniority aprenda. A su vez, el desarrollador con más experiencia puede aprender nuevas herramientas que todavía no haya implementado o aprendido. Todo el mundo es experto en algo y todos tienen algo que aprender.

Hay tres maneras de emparejamiento, explica el usuario ayushharwani2011 de GeeksforGeeks (2020).

Novato-Novato. Es un emparejamiento que puede ocurrir muy raramente pero cuando sucede puede tener grandes resultados.

Experto-Novato. Cuando se juntan desarrolladores de esta manera, se puede obtener resultados significativos. El novato puede aprender muchas cosas de un experto.

Experto-Experto. Este par es una buena elección para alcanzar resultados de alta productividad. Ya que ambos son expertos se puede conseguir resultados eficientes.

La técnica previamente mencionada –pair-programming– va a permitir al buddy insertar al nuevo empleado –en caso que sea desarrollador– en las tareas diarias. Lo ayudará a conocer las APIs que tiene el equipo y empezar a conocer el lenguaje –sea que lo conozca o no–, a familiarizarse cómo están ordenados por dentro, cómo se realizan las pruebas –más conocido como testing–, qué frameworks o librerías utilizan, entre otros conocimientos adquiridos.

Hoy en día los equipos trabajan de manera agile –ágil–, metodologías que trabajan de esta manera y las más conocidas actualmente, son Kanban y Scrum. Existen otros métodos además de los dos anteriormente mencionados, todos ellos tienen cuatro puntos clave.

Esos cuatro puntos son el proceso de diseño iterativo, el compromiso continuo de los interesados, el objetivo de software confiable y de calidad, y el desarrollo en ciclos cortos (hasta un mes) que permiten entregar software regularmente (Leybourn, 2013).

En este sentido, la metodología agile es apropiada en contextos donde los resultados son desconocidos y donde la entrega de resultados no puede estar controlada completamente (Leybourn, 2013)

La técnica de Scrum está basada principalmente en equipos y con roles asociados definidos, eventos, artefactos y reglas. Los tres principales roles que podemos encontrar en el Scrum son el Product Owner que es quien representa a los interesados del desarrollo, el Scrum Master que es quien administra el equipo y el proceso de Scrum, y por último el equipo,

cerca de 7 personas que son quienes desarrollan el software (Leybourn, 2013).

Cada proyecto se entrega de manera flexible y de manera iterativa donde al final de cada sprint de trabajo hay un entregable tangible para mostrárselo al interesado (Leybourn, 2013)

El siguiente gráfico muestra como es el proceso iterativo de Scrum.

Figura 1. Proceso de Scrum (Leybourn, 2013).

Los siguientes temas de los que se habla corresponden a las herramientas de trabajo que el buddy debe saber y que posiblemente el nuevo desarrollador tenga conocimientos. Se explican en general para que se tenga una noción de lo que puede empezar a hablarse con el buddy en caso que el nuevo empleado desconozca. A su vez se explican algunas que el TL es quién debe estar más al tanto en cuanto a profundidad de los temas –escalabilidad horizontal y vertical, base de datos, patrón de diseño, microservicios, por nombrar algunas– para lograr una optimización de los recursos y lograr aplicaciones más eficientes.

Git es el sistema de control de versionado más común utilizado. Marca los cambios hechos en un archivo, así se puede tener registro de lo que se hizo, y se puede revertir cambios a versiones específicas siempre que necesites. A su vez, Git provee colaboración más fácil, permitiendo

múltiples cambios para que se fusionen en un solo archivo (Noble Desktop, 2021).

Git es un software que corre localmente. Tus archivos y el historial son guardados en tu computadora. También puede utilizarse un host para resguardar una copia online de tus archivos y tu historial. Teniendo todo centralizado en un lugar donde se puede subir tus cambios y descargar los cambios de otros, permitiendo la colaboración más fácil entre desarrolladores. Git puede automáticamente combinar cambios, así dos personas pueden trabajar en distintas partes del mismo archivo y luego combinar esos cambios sin perder el trabajo de uno o el otro (Noble Desktop, 2021).

Un repositorio de Github es una carpeta con todos los archivos necesarios del proyecto, incluyendo los archivos que marcan todas las versiones de tu proyecto así se puede revertir los cambios en caso que se haya cometido un error. Github también muestra quién puede colaborar y cómo (Guthals & Haack, 2019).

“Slack es una aplicación de mensajería para empresas que conecta a las personas con la información que necesitan. Slack transforma la manera en que se comunican las organizaciones reuniendo a las personas para que trabajen como un equipo unificado.” (Slack, s.f.).

Los siguientes son los beneficios de la aplicación.

Conectado. “Envía mensajes a cualquiera que esté dentro o fuera de tu organización y colabora tal como lo harías en persona. Las personas pueden trabajar en espacios dedicados denominados canales que reúnen a las personas y a la información adecuadas.” (Slack, s. f.).

Este beneficio es el más importante dentro del área empresarial, ya que es la finalidad última de la aplicación, por qué se creó Slack básicamente. Para poder acceder a los canales dentro de la empresa se requiere autorización por parte de los administradores.

Flexible. “Trabajo asincrónico. Cuando el trabajo se organiza en canales, no importa tu ubicación, zona horaria o actividad. Puedes acceder a la información que necesitas a la hora que desees" (Slack, s. f.).

Este beneficio se utiliza mucho dentro de MercadoLibre, en especial para las guardias ya que dentro de un equipo se crea un canal dedicado a las alertas y será complementario a la aplicación Opsgenie[2] para poder responder de una manera rápida ante cualquier falla que presente el sistema.

Inclusivo. Todos los miembros de una organización tienen acceso a la misma información que se puede buscar y está compartida. Cuando los equipos trabajan juntos en canales, la información se puede compartir con todos a la vez, lo que ayuda a que los equipos se mantengan alineados y tomen decisiones más rápidamente (Slack, s. f.).

Este es otro beneficio a destacar ya que en la sección del buscador se puede buscar fácilmente un grupo o un mensaje dentro de un grupo de donde se esté hablando el tema de interés y poder insertarse en ese grupo sin necesidad de autorización a no ser que esté en privado el grupo y no se pueda ver, por lo que ahí si requiere avisar a una persona que se encuentre dentro.

Jira “se ha convertido en una potente herramienta de gestión de trabajo para todo tipo de casos de uso, desde la gestión de requisitos y casos de prueba hasta el desarrollo de software ágil.” (Atlassian, s.f.).

> En el caso de los equipos que usan metodologías ágiles, Jira Software proporciona tableros de scrum y kanban listos para usar. Los tableros son centros de gestión de tareas, donde estas se asignan a flujos de trabajo personalizables. Asimismo, los tableros ofrecen transparencia sobre el trabajo en equipo y visibilidad del estado de cada elemento de trabajo. Las funciones de seguimiento del tiempo y los informes de rendimiento en tiempo real (diagrama de trabajo pendiente o de trabajo completado, informes de sprints, gráficos de velocidad) permiten a los equipos supervisar de cerca su productividad con el paso del tiempo.

[2] La web oficial de Opsgenie es [https://www.atlassian.com/es/software/opsgenie]

Jira Software es compatible con cualquier metodología ágil de desarrollo de software (Atlassian, s.f.).

Los compiladores que usan los lenguajes de programación toman todo el programa como entrada y lo traducen a un ejecutable binario en varios pasos.

Solo podemos arrancar el ejecutable en la computadora donde lo compilamos. Eso es porque el código binario depende del hardware y no de la portabilidad.

El proceso de compilación sólo requiere realizarse una vez. Luego, podemos ejecutar cuantas veces queramos el código binario.

Debido a que los compiladores procesan todos los programas, son capaces de capturar algunos errores y advertirnos. Esos son errores de tipeo y sintaxis. La compilación falla si están presentes (Baeldung, 2021).

En contraste, los lenguajes que utilizan intérpretes, leen y ejecutan el programa instrucción por instrucción. Después de leer, se traduce cada instrucción en código binario y se ejecuta.

A diferencia de los compiladores, los intérpretes no producen un archivo binario ejecutable. Cada vez que se arranca el programa, se ejecuta el intérprete.

Esa es la razón por la que tiene que estar presente en la RAM de la computadora cuando corremos el programa. En contraste con los intérpretes, necesitamos compilar durante la compilación.

Por otro lado, a diferencia de los compiladores, los intérpretes capturan los errores en tiempos de ejecución (Baeldung, 2021).

Compilador	Intérprete
Procesa los programas de una	Procesa los programas una instrucción por vez
Traduce los programas a código binario de máquina	Ejecuta el programa cargando y raduciendo línea por línea en el momento
Se necesita solo después de que el programa se completó	Arranca cada vez que el programa se ejecuta
Permite la detección de algunos errores previo a la ejecución	Todos los errores son detectados durante la ejecución
No necesita estar presente en la RAM antes de la ejecución	Necesita permanecer en la RAM durante la ejecución del programa
Programas compilados generalmente se ejecutan más rápido	Los programas interpretados generalmente son más lentos

Tabla 1. Nota. Fuente: Baeldung (2021)

Postman[3] es una API (interfaz de programación de aplicaciones) que construye, testea y modifica APIs. Casi cualquier funcionalidad que el desarrollador necesite está encapsulada en esta herramienta. Es usada por desarrolladores para desarrollar sus propias APIs de manera fácil y simple. Tiene la habilidad de realizar diferentes tipos de solicitudes HTTP (GET, POST, PUT, PATCH), guardando entornos para su uso, convirtiendo la API para codear en diferentes lenguajes (como JavaScript, Python) (Hooda, 2021).

El deployment –no tiene traducción al español en IT– de software incluye todos los pasos, procesos y actividades que se requiere para hacer

[3] La web oficial de Postman es [https://www.postman.com/]

que un software o actualizaciones estén disponibles para los usuarios. Hoy en día, los desarrolladores hacen deploy de actualizaciones, parches o nuevas aplicaciones combinando procesos manuales y automatizados. Algunas de las actividades de deployment incluyen el lanzamiento del software, instalación, testeo, deployment y el monitoreo de performance (sumo logic, s. f.).

Un framework habilita un entorno de código que contiene librerías de bajo nivel para enfrentar problemas de código convencional. El objetivo del framework es entregar el desarrollo de una aplicación más rápido. Esto incluye todo lo que necesitamos para construir aplicaciones a gran escala, tal como plantillas basadas en las mejores prácticas. Internamente, un framework contiene gran cantidad de librerías que proveen al desarrollador funcionalidades ya armadas, las cuales ayudan a desarrollar una aplicación sin necesidad de tener una gran cantidad de conocimiento para codear (Roy, 2022).

Una librería es una colección de código reusable, compilado y testeado que puede facilitar la automatización o el aumento de las funcionalidades de una aplicación. Está diseñado para soportar tanto el código del desarrollador como el del compilador durante el proceso de construcción y ejecución de la aplicación. Una librería implementa muchas funciones, variables, y parámetros (Roy, 2022).

La diferencia técnica entre framework y librería radica en lo que se llama inversión de control.

Cuando se usa una librería, el desarrollador está a cargo del flujo de la aplicación. Elige cuándo y dónde llamar a la librería. En cambio, cuando usas un framework, el framework es quien está a cargo del flujo. Provee algunos lugares donde el desarrollador puede insertar el código, pero ejecuta el código que el desarrollador insertó tanto como necesita (Wozniewicz, 2019).

Un patrón de diseño es una solución repetible general a un problema común en diseño de software. A su vez, no es un diseño terminado que directamente se transforma en código. Es una descripción o una plantilla que

se aplica para resolver un problema y puede ser usado en distintas situaciones.

Un patrón de diseño puede incrementar la velocidad de los procesos de desarrollo suministrando paradigmas de desarrollo testeado y probado. Reutilizando patrones de diseño se pueden prevenir problemas sutiles que pueden causar mayores problemas y ayuda a mejorar la lectura de código.

A menudo, la gente sabe aplicar ciertas técnicas de diseño de software a ciertos problemas. Estas técnicas son difíciles de aplicar a problemas de más alcance. Los patrones de diseño proveen soluciones generales, documentadas en un formato que no requieren específicamente que estén ligados a un problema en particular.

Adicionalmente, los patrones permiten a los desarrolladores comunicarse usando nombres conocidos y entendibles para interacciones de software. Los patrones de diseño común pueden mejorarse con el tiempo, haciéndolos más robustos que los diseños ad-hoc (SourceMaking, s.f.).

Los microservicios son un enfoque organizacional y de arquitectura para el desarrollo de software donde el software está compuesto en pequeños servicios independientes que se comunican a través de APIs. Estos servicios son propios de equipos autónomos y pequeños.

La arquitectura de los microservicios hace que las aplicaciones sean más fáciles de escalar y más rápidas para desarrollar, permitiendo la innovación y aceleración para nuevas funcionalidades (AWS, s.f.).

La diferencia entre arquitectura monolítica y arquitectura de microservicios es que con el primero, todos los procesos están emparejados estrechamente y se ejecutan como un servicio único. Esto significa que si un proceso de una aplicación experimenta un pico de demanda, la arquitectura entera debe escalar. Agregando o mejorando funcionalidades de la aplicación monolítica se convierte más complejo a medida que el código base crece. Esta complejidad limita la experimentación y hace difícil implementar nuevas ideas. Las arquitecturas monolíticas añaden riesgo para la disponibilidad de la aplicación porque muchos procesos dependientes y

emparejados estrechamente incrementan el impacto de una falla de un proceso único.

En contraste, con la arquitectura de microservicios, una aplicación es construida como componentes independientes que ejecutan cada proceso de aplicación como un servicio. Estos servicios se comunican vía una interfaz bien definida usando APIs ligeras. Los servicios son construidos para capacidades empresariales y cada servicio lleva a cabo una función única. Porque son ejecuciones independientes, cada servicio puede ser actualizado, deployado y escalado para cumplir las demandas de funciones específicas de una aplicación (AWS, s.f.).

Los microservicios son autónomos y especializados. La primera característica hace referencia a que cada componente de servicio puede ser desarrollado, deployado, operado y escalado sin afectar las funciones de otros servicios. Los servicios no necesitan compartir nada de su propio código o implementación con los otros. Cualquier comunicación entre componentes individuales sucede via APIs bien definidas.

Por otro lado, la segunda característica se refiere a que cada servicio es diseñado para un set de capacidades y concentrado en solucionar un problema específico. Si los desarrolladores contribuyen más código a un servicio con el tiempo y el servicio se vuelve complejo, puede separarse en pequeños servicios (AWS, s.f.).

Los beneficios de los microservicios son los siguientes.

Son ágiles, los microservicios alientan una organización de equipos pequeños e independientes que tomen posesión de sus servicios. Los equipos actúan dentro de un contexto pequeño y entendible, y son empoderados a trabajar más independientes y más rápidos. Esto acorta el tiempo del ciclo de desarrollo. Se beneficia significativamente del rendimiento agregado de la organización.

Escalabilidad flexible. Los microservicios permiten que cada servicio sea independientemente escalable para satisfacer con la demanda de las funcionalidades de la aplicación que soportan. Esto habilita que los equipos adecúen el correcto tamaño de la infraestructura que se necesita, precisar la medición del costo de una funcionalidad, y mantener disponibilidad si un servicio experimenta un pico de demanda.

Fácil deployment. Los microservicios habilitan integración continua y entrega continua, haciendo fácil intentar nuevas ideas y revertirlas si algo no funciona. El costo bajo de fallas permite la experimentación, haciendo más fácil actualizar código, y acelerar tiempo de mercado para nuevas funcionalidades.

Libre de tecnología. La arquitectura de microservicios no sigue un enfoque “talla única para todos”. Los equipos tienen la libertad de elegir la mejor herramienta para solucionar sus problemas específicos. Como consecuencia, los equipos construyen microservicios pueden elegir la mejor herramienta para cada trabajo.

Código reusable. Dividiendo software en pequeños y módulos bien definidos permite a los equipos usar funciones para múltiples propósitos. Un servicio escrito para una cierta función puede ser usado como un bloque para otra funcionalidad. Esto permite que una aplicación arranque por sí misma, así como los desarrolladores pueden crear nuevas capacidades sin crear código desde el principio.

Resiliencia. Los servicios independientes incrementan la resistencia de una aplicación contra los fallos. En una arquitectura monolítica, si un solo componente falla, puede causar que la aplicación entera falle. Con microservicios, las aplicaciones controlan el servicio total de fallos degradando la funcionalidad evitando que se rompa la aplicación entera (AWS, s.f.).

Datadog es una herramienta que permite monitorear la infraestructura en la nube, el hosteo en Windows y Linux, procesos de sistema, funciones serverless –sin servidor–, y aplicaciones basadas en la nube. Puede ser utilizado para visualizar datos, explorar métricas, administrar logs, y realizar otras tareas diferentes.

Datadog permite recolectar métricas y juntar en tiempo real visiones profundas acerca de la infraestructura aplicada. Aquí están los principales casos de usos de la aplicación:

Los profesionales IT pueden crear, editar y administrar alertas y notificaciones acerca de su infraestructura.

Las organizaciones pueden utilizar la herramienta APM –Application Performance Monitoring– para reducir la latencia y eliminar errores.

Pueden testear entornos de producción y performance.

Pueden configurar múltiples integraciones que recolectan métricas, seguimientos, y logs para enviar información a la plataforma.

Pueden usarlo como plataforma de seguridad para detectar amenazas y configuraciones erróneas en su infraestructura.

Si utilizan Jenkins, que es un server automatizado para el deploy de software, la aplicación puede ayudar a visualizar métricas de trabajo de Jenkins y pasos de ejecución (Sagar, 2022).

New Relic es un software de observabilidad. Los equipos lo usan para monitorear la performance de sus aplicaciones e infraestructura.

Parte de desarrollar una aplicación exitosa tiene que ver con mantenerla funcionando luego de construirla.

Las dos áreas principales que los equipos necesitan visualizar es la capa de aplicación –APM– y la capa de infraestructura –los servidores– (Justin, 2022).

New Relic provee una serie de APIs para la recolección de información de esas capas, visualizando esa información, y profundizando en por qué las cosas fueron mal.

Otras buenas características que provee New Relic son monitoreo de navegación de producto que ayuda al seguimiento de métricas comunes como la carga de página, el tiempo gastado de página, estabilidad visual y errores. IA para monitoreo que automáticamente analiza la performance de la información para encontrar errores comunes y las causas de raíz. Por último, workflows en el IDE, New Relic compró CodeStream en octubre de 2021 y lo integró en su paquete rápidamente. La esencia es básicamente mover un montón de cosas como si se estuviera haciendo en GitHub o la interfaz de New Relic en el entorno de desarrollo de tu elección (Justin, 2022).

Opsgenie notifica a las personas, reduce el tiempo de respuesta y evita el estrés de alerta.

Opsgenie es una plataforma de administración de incidentes modernos que aseguran que los incidentes críticos nunca se pierdan, y que se tomen acción por las personas correctas en el tiempo más corto posible. Opsgenie recibe alertas del sistema de monitoreo y aplicaciones personalizadas y categorías de cada alerta basada en importancia y tiempo.

En horarios de guardia, Opsgenie asegura que las personas correctas sean notificadas a través de múltiples canales de comunicación incluyendo llamadas, emails, SMS, y mensajes en los celulares. Si una alerta no es reconocida –cuando te suena una alerta hay que clickear un botón que reconoces la alerta–, Opsgenie automáticamente la escala, asegurando que el incidente tenga la atención requerida (Atlassian, s.f.).

Swagger permite describir la estructura de tus APIs así las computadoras pueden leerlas. La habilidad de APIs de describir su propia estructura es la raíz de todo lo asombroso en Swagger. Leyendo la estructura de tu API, automáticamente podemos construir documentación interactiva. También podemos generar automáticamente librerías para la API en varios lenguajes y explorar otras posibilidades como pruebas automatizadas. Swagger hace esto preguntándole a tu API si devuelve un YAML o JSON que contenga una descripción detallada de tu API entera. Este archivo es esencialmente una lista de recursos de tu API que se adhiere a la especificación OpenAPI.

Se puede escribir una especificación Swagger manualmente para tu API, o que se genere automáticamente desde las anotaciones en tu código fuente (Swagger, s.f.).

La prueba de software es un método para verificar si el producto de software actual cumple los requerimientos esperados y asegura que el producto está libre de defectos. Incluye ejecución de los componentes del sistema/software usando herramientas manuales o automatizadas para evaluar una o más propiedades de interés. El propósito de las pruebas de software es identificar errores, huecos o requerimientos faltantes contrastando con los requerimientos actuales (Hamilton, 2022).

Las pruebas unitarias son típicamente pruebas automatizadas que verifican que cada unidad –o partes de código aisladas– funcionen como el desarrollo quiere.

Las pruebas unitarias son individualmente realizadas, a menudo conocidos como “casos de prueba” que consisten en segmentos de código que trabajan juntos para realizar una función específica. Cada prueba unitaria evalúa el código escrito y asegura que se alinee con lo que la función dice.

Sin embargo, las pruebas unitarias son más estructurales, quiere decir que no interactúan con APIs subyacentes. Además, no evalúan la interfaz de usuario o cualquier función de usuario final.

A diferencia de los otros tipos de pruebas, las unitarias no incluyen al usuario final como objetivo, lo que hace que cada prueba unitaria sea única. Mientras que las pruebas no se enfocan en usabilidad u otro tipo de aspectos no funcionales de una aplicación, todavía sirven como autenticación genuina de que se cumplan los requerimientos de usuario.

La calidad de las pruebas unitarias depende de la habilidad de prever e implementar casos correctamente que deberían estar en el conjunto de pruebas. La práctica común es agregar casos de pruebas que se refieren a errores específicos identificados durante el uso de producción. Tiene sentido implementar pruebas unitarias en partes críticas de la aplicación como el login, pagos, entre otros.

Además, las pruebas unitarias son escritas y leídas por desarrolladores. El punto es verificar y ver si el código funciona apropiadamente, documentarlo, incrementar la calidad, reducir el costo de arreglar errores y fallas productivas.

Por último, las pruebas unitarias dejan saber a los desarrolladores si la aplicación está lista para usar o no (RevDeBug, 2021).

Una prueba de integración es una prueba modular de software. Involucra los módulos que son lógicamente integrados, y esos módulos son puestos a prueba como un grupo en vez de pruebas individuales.

Específicamente, las pruebas de integración tienen principalmente foco en la comunicación de información. Estas pruebas son críticas

considerando que cada proyecto de software consiste típicamente en múltiples módulos que son codeados por programadores.

La integración es usualmente llevada a cabo por testers –Gente que realiza las pruebas– específicos a través de dos métodos: El método bottom-up –abajo hacia arriba– y el método top-down –arriba hacia abajo–. En otras palabras, cada tester comienza en un segmento de módulo diferente funcionando del front al back y de back al front para asegurar consistencia (RevDeBug, 2021).

Un IDE –Entorno de Desarrollo Integrado– permite a los programadores consolidar los diferentes aspectos de escribir un programa de computadora.

IDEs incrementan la productividad de los programadores combinando actividades comunes de escritura de software en una sola aplicación: editando código fuente, construyendo ejecutables, y debuggeando –práctica para encontrar errores– (Codeacademy, s.f.).

Una base de datos es una colección de información organizada, así puede ser fácilmente accesible y administrada.

Se puede organizar información en tablas, filas, columnas y con índices para hacerlos más fácil al buscar información relevante.

Los administradores de base de datos crean una de manera que un set de softwares acceda a la información de todos los usuarios.

El propósito principal de una base de datos es operar una larga carga de información almacenando, recuperando y administrando información.

El lenguaje SQL –Structured Query Language– es usado para operar la información almacenada en la base de datos. Se utiliza una estructura cilíndrica como imagen para representar una base de datos (JavaTpoint, s.f.).

Existen dos tipos de base de datos, las relacionales y las no relacionales.

Una base de datos relacional almacena información en tablas. A menudo, estas tablas comparten información entre ellas, causando una

relación entre ambas. De aquí es donde las bases de datos relacionales tienen el nombre.

Una tabla usa columnas para definir la información que es almacenada y filas para la información actual. Cada tabla tendrá una columna que debe tener valores únicos –conocidos como clave primaria–. Estas columnas pueden ser usadas en otras tablas, si las relaciones son definidas entre ellas. Cuando una clave primaria de una tabla es usada en otra tabla, esta columna en la segunda tabla es conocida como clave foránea (MongoDB, s.f.).

En cambio, una base de datos no relacional, algunas veces llamada NoSQL, es cualquier tipo de base de datos que no usa tablas, campos, ni columnas con información estructurada. Fueron diseñadas con la nube en mente, haciéndolos de mejor utilidad al escalar horizontalmente.

CARACTERÍSTICA	NO RELACIONAL	RELACIONAL
DISPONIBILIDAD	Alta	Alta
ESCALA HORIZONTAL	Alta	Baja
ESCALA VERTICAL	Alta	Alta
ALMACENAJE DE DATOS	Optimizado para gran volumen de información	Media a mucha información
PERFORMANCE	Alta	Baja a media
SEGURIDAD	Media	Alta
COMPLEJIDAD	Baja	Media
FLEXIBILIDAD	Alta	Baja

Tabla 2. Nota. Fuente: Blancarte, Oscar (2017)

La escalabilidad es “la capacidad del software para adaptarse a las necesidades de rendimiento a medida que el número de usuarios crece, las transacciones aumentan y la base de datos empieza a sufrir degradamiento del performance por las cargas crecientes” (Blancarte, 2017).

Podemos distinguir dos tipos de escalabilidad, vertical y horizontal. En cuanto al primero podemos especificar que es la manera de “crecer el hardware de uno de los nodos, es decir, aumentar el hardware por uno más potente” (Blancarte, 2017). Este es el tipo de escalabilidad más simple ya

que no requiere esfuerzo y no tiene impacto en el software, requiere solamente “respaldar y migrar los sistemas al nuevo hardware” (Blancarte, 2017), en caso que haya que reemplazar el hardware por la totalidad en vez de añadir partes. En el siguiente gráfico se puede ver de manera visual.

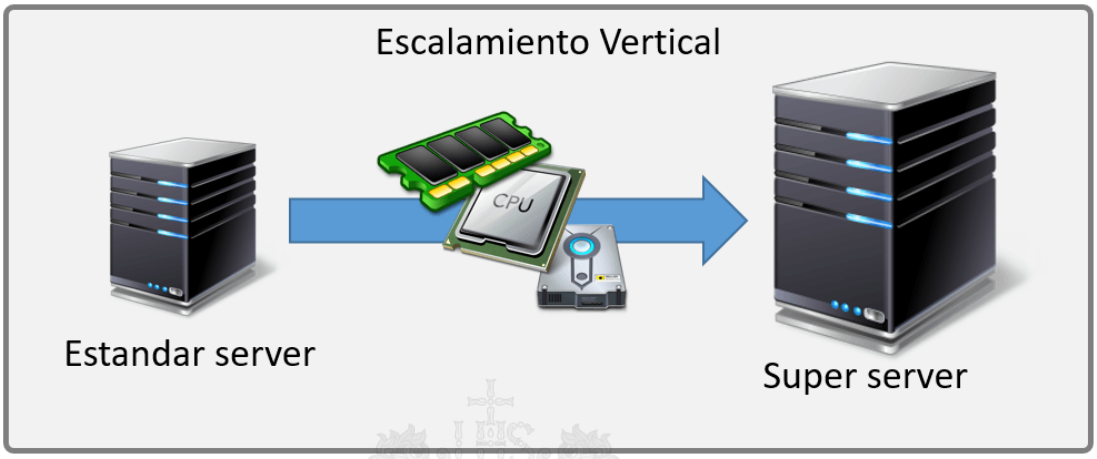

Figura 2. Escalamiento Vertical (Blancarte, 2017).

La desventaja principal de este tipo de escalamiento es el propio hardware, ya que llega un momento que por tema de compatibilidad no se podrá combinar nuevo hardware con el viejo existente que tengamos. Por lo que a futuro tendremos que reemplazar todo el hardware. Sin embargo, “podemos combinar con el escalamiento horizontal para obtener mejores resultados.” (Blancarte, 2017).

Ventajas	Desventajas
Cambio en el hardware, sin problemas para las aplicaciones.	Limitado por hardware
Fácil implementación	Una falla en el servidor implica que la aplicación se detenga
Solución rápida y económica	No soporta alta disponibilidad
	Hacer un cambio total de hardware es más caro.

Tabla 3. Nota. Fuente: Blancarte, Oscar (2017)

En cambio, la escalabilidad horizontal es más potente pero más complicada de implementar. Esto implica “tener varios servidores –conocidos como nodos– trabajando como un todo. Se crea una red de servidores –conocida como cluster–, con la finalidad de repartirse el trabajo entre todos

nodos del cluster, cuando el performance del cluster se ve afectada con el incremento de usuarios, se añaden nuevos nodos al cluster" (Blancarte, 2017) y así sucesivamente.

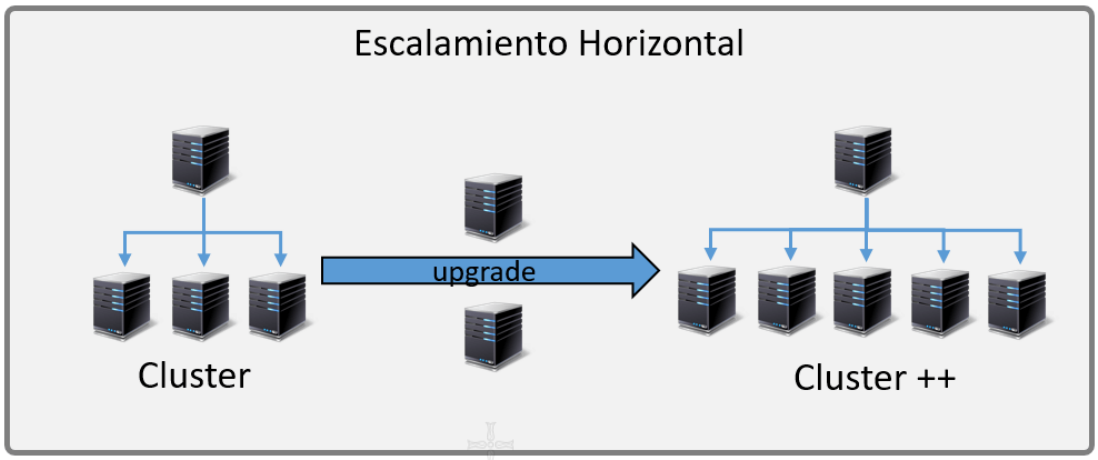

Figura 3. Escalamiento Horizontal (Blancarte, 2017).

La manera de aplicar la escalabilidad horizontal es tener "un servidor primario desde el cual se administra el clúster. Cada servidor del clúster deberá tener un software que permite integrarse al clúster [...] y sobre estos se montan las aplicaciones que queremos escalar." (Blancarte, 2017).

Ventajas	Desventajas
Crecimiento infinito	Requiere mucho mantenimiento
Combinarse con escala vertical	Difícil de configurar
Soporta alta disponibilidad	Requiere grandes cambios en las aplicaciones
Si falla un nodo, los demás siguen trabajando	Requiere infraestructura más grande
Soporta balanceo de cargas	

Tabla 4. Nota. Fuente: Blancarte, Oscar (2017)

Metodología

Para el estudio de la presente investigación, además de las distintas consultas bibliográficas, se llevó a cabo un cuestionario autoadministrado hecho en Google Form[4].

Entendiendo al universo como “el conjunto de población para la cual tiene validez el conocimiento producido por la investigación. Son todos los miembros de cualquier clase bien determinada de personas, eventos u objetos” (Saavedra R., 2001, p. 45). En este caso, se toma como universo las personas que trabajan dentro de empresas se aplica el Buddy System.

La muestra es una parte del universo y “debe ser representativa de los sujetos que componen la población y suficientes para que los resultados en efecto puedan generalizarse a toda la población o universo” (Saavedra R., 2001, p. 45). Dado lo anterior, nuestra muestra son las personas que trabajan dentro de MercadoLibre.

“Las Unidades de Análisis son los elementos menores y no divisibles que componen el universo de estudio de una investigación. Sobre dichos elementos se estudia el comportamiento de las variables” (Abritta, 2014, p. 2). De esta forma, nuestras unidades de análisis son las personas que se encuentran trabajando en equipos de diseño con sus respectivas variables.

Las variables son los aspectos de nuestras unidades de análisis que tendrán un valor determinado. Estas variables que nos ayudan a confirmar o refutar nuestra hipótesis son si el buddy de la persona –nuestra unidad de análisis– fue del mismo equipo y si fue del mismo rol que cuando ingresó.

Teniendo en cuenta las dos variables previamente mencionadas, las mismas tendrán un valor. El valor es definido por el autor Abritta como “diferentes opciones o alternativas que presenta la variable y adopta alguna unidad de análisis y se puede expresar cualitativamente a través de una clasificación por ausencia y presencia, por jerarquía u orden o sino cuantitativamente, es decir, a través de magnitudes”. Considerando la definición anterior de Abritta, tenemos variables cualitativas y dentro de las mismas son dicotómicas. Las variables dicotómicas son aquellas que “permiten tomar dos valores posibles” (QuestionPro, s.f.)

[4] https://docs.google.com/forms/d/1f-4jBwa6YL7DH6wOtJgAzIzbmEIxRFvZqh7Ca7SCsL4

Se define indicador como “un signo (propiedad, atributo, variable) mediante el cual nos aproximamos al conocimiento de ciertas características de un objeto que no se pueden medir directamente (de aquí que se hable de inferencia)” (Abritta, 2014, p. 3). Para analizar los resultados que llevan a refutar o confirmar la hipótesis, la segunda pregunta de la sección 1, nos indica si tuvo o no un buddy que en caso afirmativo nos demuestra que tuvo experiencia con el Buddy System. Mientras que la sección 2, hace preguntas en referencia a las variables en cuestión con la experiencia que tuvo. Tomando en cuenta el resultado de la segunda sección, la persona puede responder las preguntas de las dos variables con un enfoque basado en la experiencia. A su vez, se hace una pregunta libre dentro de la variable para obtener un valor agregado de por qué consideran que el buddy tiene que ser o no del mismo equipo y/o mismo rol.

El siguiente formulario fue el utilizado para recolectar los datos pertinentes de la muestra. Se brinda una imagen como referencia.

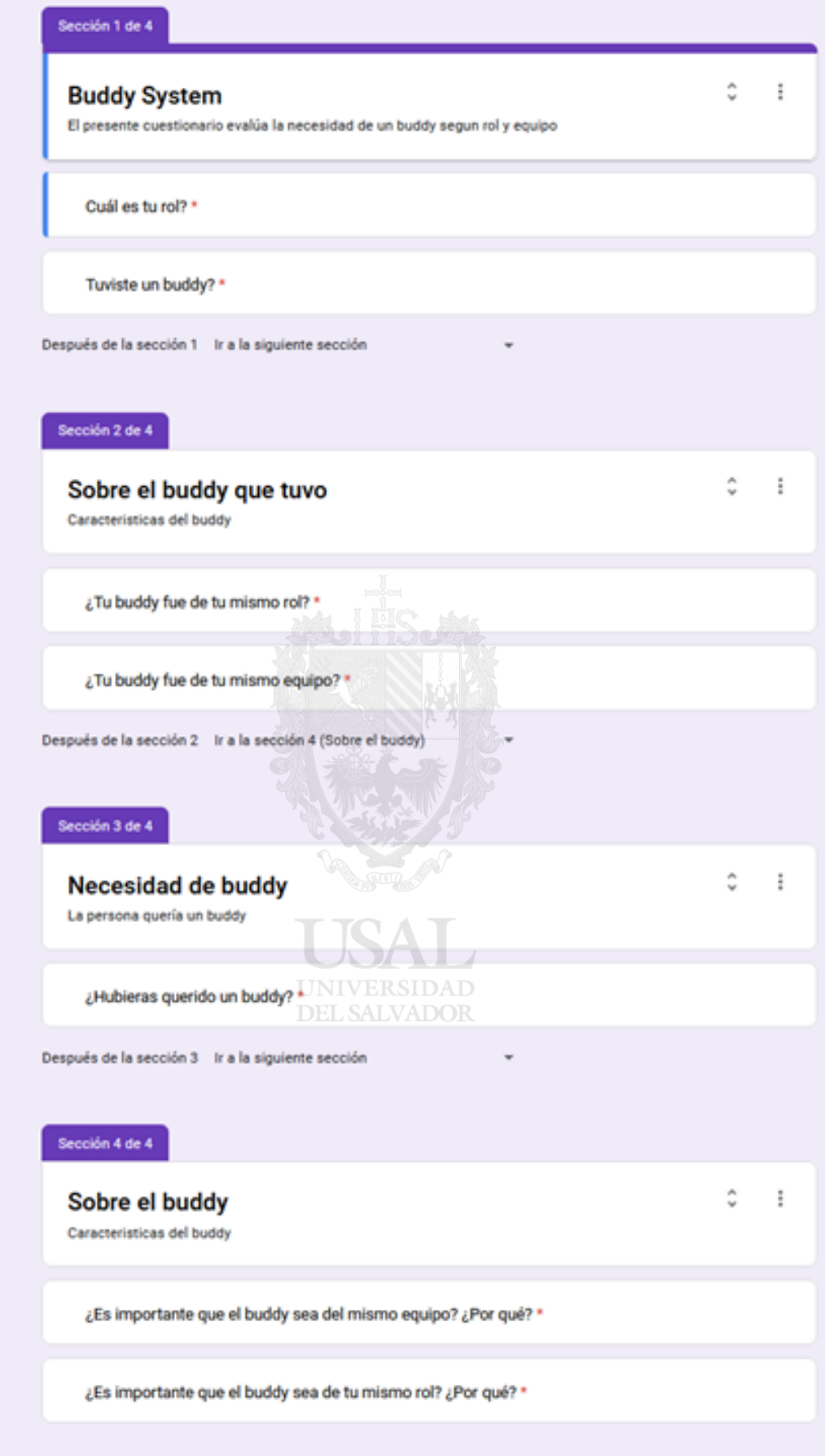

Figura 4. Cuestionario Buddy System (Abarquez Mendoza, 2022).

En caso que se decida realizar un experimento más acotado y se decida realizar un análisis solo con la experiencia, es necesario realizar un formulario en el cual solamente los que respondan acertadamente que tuvieron un buddy también puedan responder las dos variables a analizar –sección 4 del formulario–. Por otro lado, si la intención es realizar un experimento acotado y más abarcativo que sólo tengan en cuenta a aquellos que no tuvieron un buddy, se puede realizar un formulario en el cual solo se incluya la sección 4. Por último, se puede realizar el mismo experimento que toma en cuenta tanto la experiencia de haber tenido un buddy como aquellos que no tuvieron.

La encuesta autoadministrada fue compartida por un canal de Slack donde se encuentran quienes fueron algunos de mis compañeros de bootcamp –un grupo donde hay 14 personas–, y en el grupo de Whatsapp que tenemos en común. A su vez también fue compartida por el canal de Slack donde se encuentran otros equipos bajo la misma dirección de manager, donde hay actualmente 29 personas, sin embargo, en el momento en que se compartió y se aceptaban respuestas había 27 personas. El formulario quedó a disposición 2 semanas con el fin de obtener cierta cantidad de respuestas.

En base a lo anterior, podemos afirmar que 41 personas recibieron el cuestionario a completar, sin tener en cuenta las 2 personas nuevas que ingresaron posteriormente. Teniendo en cuenta el total previamente mencionado y que 16 fueron las que completaron el cuestionario, se puede apreciar que el 39% enviaron sus respuestas frente al Buddy System. Mientras que hubo un rechazo del 61% a completar el mismo.

Una vez terminado el análisis sobre cuántas personas fueron quienes completaron el formulario. Procedemos a analizar los resultados que hubo en cada pregunta con sus respectivas variables.

En la primera parte podemos destacar, que más de la mitad de los profesionales tuvieron buddy. El 62.5% implica que 10 personas tuvieron buddy al entrar a la empresa y 6 personas no.

Ahora bien, evaluando la imagen anterior y para que se entienda cómo son los roles dentro de MercadoLibre, tomemos en cuenta la siguiente tabla y ordenada por seniority ascendente previo al análisis.

Rol MercadoLibre	Rol Cuestionario
Expert	Expert FBM Inbound
Project Leader	Project Leader
Project Leader	Líder de Proyecto
Sr. Software Engineer	Desarrollador Sr
Software Engineer	Ssr
Software Developer	Desarrollador de Software

Tabla 5. Nota. Tabla realizada en el contexto de este proyecto.

Como podemos observar, de las 6 personas tenemos 1 Expert, 2 PL, 1 Sr, 1 Ssr y 1 Developer. Es decir, cualquiera que entre a MercadoLibre puede tener un buddy o no. Eso puede deberse a que cada equipo se comporta de manera diferente, agregando que recién puede estar formándose y no hay gente para asignar.

Considerando a los que tuvieron buddy, podemos ver que 60% tuvieron uno del mismo rol, 6 personas, y 40% de diferente, las restantes 4. A su vez 90% tuvieron buddy del mismo equipo, 9 personas, y 10% o 1 persona tuvo un buddy de un equipo diferente.

Por último, y una de las más destacadas para la presente tesis, tenemos el porcentaje que hubiera querido un buddy. Nos da el 100%, es decir las 6 personas que no tuvieron buddy. Desglosando las diferentes variables que se les preguntó y analizándolas, esta es la pregunta a considerar ya que no tuvieron buddy y por algún motivo lo vieron necesario.

En resumen, podemos afirmar que más de la mitad de los participantes tuvieron un buddy del mismo equipo y del mismo rol. Sin embargo, nos falta el último detalle que sería la experiencia que tuvieron. Es decir, si bien los emparejaron así, no sabemos si el Buddy System implementado en MercadoLibre logra ser eficiente y eficaz. Por lo que

recurrimos a ver el resultado de las dos últimas preguntas para poder afirmar o refutar nuestra hipótesis.

Considerando la tabla de respuestas completa que podemos ver a continuación, podemos destacar las siguientes cuestiones.

Figura 5. Respuestas cuestionario Buddy System (Abarquez Mendoza, 2022).

Cuál es tu rol?	Tuviste un buddy?	¿Tu buddy fue de tu mismo rol?	¿Tu buddy fue de tu mismo equipo?	¿Hubieras querido un buddy?	¿Es importante que el buddy sea del mismo equipo? ¿Por qué?	¿Es importante que el buddy sea de tu mismo rol? ¿Por qué?
Software Developer	Sí	Sí	Sí		Sí es imporatante, por que te podrá sacar dudas técnicas y explicarte la forma de trabajo, de una manera más alineada a como piensan y trabajan los compañeros de equipo.	También considero importante, por que podrá ponerse en tu lugar en muchas situaciones que se presenten durante tus primeros meses en la empresa.
Software Engineer	Sí	Sí	No		No, la idea del buddy pienso que es que te ayude a incorporarte a MeLi y sus buenas prácticas. Suficiente con que conozca el ecosistema y tenga buena onda.	Sí, me parece importante ya que facilita la resolución de problemas referidos al trabajo en particular que estas haciendo. Si es de tu mismo rol va a tener mayor conocimiento sobre las tareas que vayas a realizar.
Analista semi senior	Sí	Sí	Sí		Si es importante pues conoce la forma en la que el equipo trabaja y por ello puede dar información importante de esta índole la cual disminuye el tiempo de aprendizaje.	Es importante que pertenezca a un rol parecido para poder responder las dudas y guiarte efectivamente en tomar parte en las actividades del equipo.
Ssr	No			Sí	No es que sea importante realmente. Cuando uno recién ingresa, hay tanto que aprender, tanto con lo que ponerse al día que cualquier persona que trabaje en la empresa hace un tiempo puede servir. Luego de ese período de adaptación si pasa a ser más necesario tener a alguien del equipo como buddy ya que es el que tiene el conocimiento del negocio puntual, es el que está al día con los proyectos del team y es el que más va a poder despejar dudas certeramente.	No. De hecho, es importante creo que sea de un rol superior digamos. Que sea alguien más experimentado, con tiempo en el equipo, que pueda ayudar al ingresante a aprender todo lo que tiene que aprender de la mejor forma, lo más rápido y con la mayor seguridad posible. De esta forma, el ingresante tiene un referente sólido en quién apoyarse.
Manager	Sí	Sí	Sí		En mi opinión, es importante pero no bloqueante. Ayuda en cuanto al contexto y cercanía con los temas y proyectos en los que la persona debe enfocarse.	Similar a la anterior, yo lo veo como lo mejor, en lo posible. Podría acelerar la adaptación.
analista de software	Sí	No	Sí		Si. Porque conoce el negocio del equipo y las intricancias.	No. Idealmente mas seniority.
Project Leader	No			Sí	No, EL buddy te guia y orienta, mientras mas amplia la visión considero mejor la guia.	No, pero si de alguien que haya tenido el mismo rol en el ultimo tiempo
Cloud Software Development Analyst	Sí	No	Sí		Si, porque es el/la que te acompaña a dar los primeros pasos de tu dia a dia. Siendo del mismo equipo tiene mas idea de como van a ser esos dias para ayudarte.	No, porque puede ser alguien con mas señiority y no habria conflicto para que me ayude en mis primeros dias en la empresa/equipo
Lider de Proyecto	No			Sí	No. Porque lo importante es acompañar el crecimiento de la persona dentro del proyecto, independiente del equipo que integre, brindando las herramientas que se emplean dentro de la compañia.	No. Porque lo que se busca es orientar sobre las tecnologías y la metodología de trabajo. El equipo en su conjunto, acompaña luego el desarrollo profesional.
Expert FBM Inbound	No			Sí	Además de la ayuda para conocer las herramientas de uso general y configuraciones, sabe a quiénes involucrar a las reuniones, los canales de uso cotidiano del equipo, y todo el contexto en general estaría abocado al scope final de trabajo	No. Es una persona que ayuda en el acompañamiento inicial para la dinámica de trabajo, el contexto, y facilitar el acceso a los referentes que darían lugar a seguir creciendo en contactos y conocimientos. Idealmente debería ser una persona que si conozca las máximas o lo que se espera de mi rol.
Desarrollador de software	No			Sí	Si, porque es quien conoce las tareas especificas del equipo y puede guiarte mejor	Si, o que al menos tenga muy claro lo que se hace en tu rol
UX Writer	Sí	No	Sí		Si, porque ayuda a que te muestre la dinámica de trabajo intra-equipo, no solo aspectos de la empresa en general.	Si, en mi caso me pusieron dos buddies, uno de mi equipo y otro de mi rol (en mi equipo no hay nadie que tenga mi rol). Estuvo bueno tener un buddy con mi rol porque me explicó todo el flujo de trabajo y qué hacer en cada oportunidad o proyecto. Creo que es clave que alguien con tu mismo rol te pueda guiar.
Desarrollador SR	No			Sí	Es esencial que sea del mismo equipo por contexto de negocio y tecnico, y porque es el que te va a nivelar para que esas otras personas no carguen con tantas tareas que va a poder realizarlas el nuevo dev	Es muy importante para sacar el maximo provecho. En mi caso me ayudaron desde TL, PL, dev de back (poco) y principalmente un dev de front, siendo que soy de back. El PL y el dev de front te da contexto de negocio, pero el TL y el dev de back te dan asesoria tecnica mucho mas explicitas e importantes que se necesita dia a dia
Software Developer	Sí	No	Sí		Por las reglas del negocio, la arquitectura además de que te ayuda a integrarte más rápido	Mi buddy es software Engenier un cargo más arriba que el mío entiendo, ayuda la experiencia
software developer	Sí	Sí	Sí		Sí porque se simplifica la tarea de guiarlo a uno a traves de las funciones del equipo.	Sí porque es de mucha ayuda el poder realizar consultas sobre las tareas ademas de los funcionamientos de la empresa.
Desarrollador Backend	Sí	Sí	Sí		Yo creo que si, te da una perspectiva temprana de los problemas específicos que hay y suma información funcional que quizás se tardaría mas en adquirir	Ayuda a sumar conocimiento técnico mas rápido

Algunas personas no respondieron por sí o por no. Sin embargo, al leer las respuestas dejan en claro su postura.

Mayoritariamente respondieron a la pregunta del rol considerando el seniority. Sin embargo, se apuntaba a entender el rol como el cargo que ejerce. Algunos ejemplos dentro del área de diseño pueden ser diseñadora UX, UX writer –no existe traducción ya que ese cargo es relativamente nuevo y se lo denomina así–, desarrollador backend, desarrollador frontend. En mi experiencia pasada, una traductora fue mi buddy, mientras que yo daba mis primeros pasos como desarrollador.

Las preguntas libres que agregan ese valor añadido a las respuestas de las variables, también se les hizo a los que no tuvieron buddy y querían, debido a que saben qué tipo de preguntas le hubieran hecho a su buddy y quién sabría responderles.

Resumiendo, las respuestas en una tabla quedarían de la siguiente forma.

Rol	¿Es importante que el buddy sea del mismo equipo?	¿Es importante que el buddy sea del mismo rol?
Software Developer	Sí	Sí
Software Engineer	No	Sí
Analista semi senior	Sí	Sí
Ssr	No	No
Manager	Sí	Sí
analista de software	Sí	No
Project Leader	No	No
Cloud Software Development Analyst	Sí	No
Lider de Proyecto	No	No
Expert FBM Inbound	Sí	No
Desarrollador de software	Sí	Sí
UX Writer	Sí	Sí
Desarrollador SR	Sí	Sí
Software Developer	Sí	Sí
Software developer	Sí	Sí
Desarrollador Backend	Sí	Sí

Tabla 6. Nota. Tabla realizada en el contexto de este proyecto.

Buddy mismo equipo	Buddy mismo rol	Total
No	No	3
No	Sí	1
Sí	No	3
Sí	Sí	9

Tabla 7. Nota. Tabla realizada en el contexto de este proyecto.

En conclusión, podemos visualizar que el 56.25% está a favor de que, utilizando la metodología Buddy System, el buddy asignado al nuevo profesional sea del mismo equipo y del mismo rol. De esta manera podemos afirmar que la hipótesis es verdadera.

De otro modo, podemos ver que 13 personas responden a que mínimamente una de las dos variables, sea el buddy del mismo equipo o del mismo rol, tiene que cumplirse. Esto representa un 81% del total de personas.

A su vez podemos ver los beneficios en los cuales tiene un impacto el buddy.

- Reconocer tareas diarias del equipo
- Adquirir conocimientos técnicos
- Solucionar dudas técnicas
- Reconocer las buenas prácticas dentro del equipo
- Disminuir tiempo de aprendizaje

Pieza de Diseño

Para demostrar las implicancias de un buddy, al presente informe se le agrega un video audiovisual demostrando las cuatro situaciones que puede haber en una empresa implementando la metodología y considerando las dos variables a analizar en la hipótesis.

¿Cómo obtenemos la cantidad de situaciones a demostrar? Es una pregunta que se puede hacer la persona que quiera realizar el experimento. La fórmula para obtener la cantidad de situaciones es 2^n. La fórmula previamente dicha tiene que ver con lógica. El número 2 hace referencia a la cantidad de valores que tienen nuestras variables –sí o no–. N hace referencia a la cantidad de variables que tenemos en nuestro experimento. En este caso N equivale a 2 variables, lo que da como resultado 4 situaciones.

La primera situación que vemos en el video es que al ingresante se le asigna un buddy que no es ni del mismo equipo ni del mismo rol. Esto influye en una incorrecta utilización del Buddy System, ya que el nuevo profesional que se une al equipo puede llegar a tener más dudas que certezas. Es decir, supongamos que el ingresante, siendo un desarrollador backend, quiere saber cuáles son las herramientas tecnológicas del equipo. Ahora bien, el buddy es alguien de recursos humanos, quien le hace la inducción y le explica sobre los temas generales de la empresa. Sin embargo, a la hora de responder esta pregunta, lo más probable es que le conteste una generalización y es que las tecnologías que se utilizan en cada equipo, dependen estrictamente de lo que prefieran sus miembros. Por otro lado, imaginemos que el desarrollador backend se sienta en su escritorio, empieza a trabajar y nota algo peculiar del código. Sabe que aun teniendo dudas no va a poder preguntárselo a su buddy, recordemos que es de recursos humanos, por lo que va a preguntarle a un miembro del equipo que pueda entender el código.

Ahora, la segunda situación nos encuentra con que el buddy del profesional que ingresa es del mismo equipo, pero no del mismo rol. En este caso el buddy es un desarrollador frontend. Esto va a incrementar la

productividad desde el comienzo, ya que, retomando la primera pregunta de la primera situación, el buddy va a poder contestar cuáles son las herramientas que manejan dentro del equipo. Algunos ejemplos pueden ser los lenguajes de programación que utilizan y las herramientas colaborativas. Consideremos nuevamente la segunda duda que tenía el ingresante, esto es respecto al código. Otra vez se ve el mismo problema, el buddy no va a poder contestarle ya que, si bien es un desarrollador, no es un lenguaje de programación que entienda. Puede verse una analogía con una persona que sabe sólo árabe y la otra persona sabe sólo inglés, es como querer interpretar lo que la otra persona dice.

Otra posibilidad, una tercera situación, puede ser que sea al revés de la anterior. Es decir, el buddy sea del mismo rol, pero no del mismo equipo. En esta ocasión, la eficacia y eficiencia va a ser de un 50% de igual modo. El buddy va a poder responder preguntas más técnicas como la segunda, y menos sobre cómo se comporta el equipo.

La última posibilidad es la que se propone en la hipótesis como la manera más eficiente y eficaz de implementar el Buddy System. Haciendo referencia a que un buddy del mismo equipo y del mismo rol va a poder contestar ambas preguntas del ingresante. Esto va a aumentar la productividad rápidamente, generar confianza entre ambos y formar una alianza en los primeros meses del profesional que acaba de ingresar. Por un lado, la confianza del nuevo miembro del equipo en poder preguntar las dudas que tenga. Desde el lado del buddy, poder mejorar la comunicación, demostrar que está capacitado para tener una persona a cargo, mejorar las habilidades blandas.

El proceso para diseñar fue extenso, con cambios de ideas una tras otra. En el blog de Domestika se nombran cinco etapas en el proceso y la primera es la inspiración. “La inspiración es artística e intelectual. Tiene que ver con los sentidos pero también con buscar información que pueda servir.” (Tempone, 2020).

Por el lado inspiracional, lo que me llevó a realizar esta pieza es la demostración de los aspectos beneficiosos de tener un buddy. A través de la experiencia que tuve en dos ocasiones y haciendo uso de mis sentidos para

absorber o no el conocimiento compartido por ellos, fue una motivación de realizar tanto la pieza artística como esta tesis.

La segunda etapa de la que habla Tempone (2020) tiene que ver con la investigación, “se lleva adelante una acumulación y organización sistemática de información”. En cuanto a este paso, la investigación permite plasmar la intención de la experiencia obtenida. Es decir, recabar la información desde el Buddy System hasta la experiencia permite poner en la pieza artística la intención de demostrar los beneficios, el objetivo, la forma de aplicarlo, la manera de utilizar dicha metodología como parte del proceso de inducción.

Como tercera etapa del proceso de diseño, encontramos la ideación. Respecto a ello, Tempone (2020) explica que no solo es “simplemente tener una idea fantástica sino más de diez buenas ideas y cien malas ideas”. En sintonía con lo dicho al principio de las etapas, la idea tuvo varias mutaciones a lo largo del proceso. Sin embargo, la que se decidió es aquella que represente visualmente en cómo un buddy puede mejorar la productividad de la persona que entra al equipo. Adicionalmente, se representa los casos en que fallaría una mala aplicación de la metodología.

La cuarta etapa se denomina verificación, Tampone (2020) nos dice que en ella “contrastamos nuestras ideas del proceso de ideación con lo que descubrimos en la investigación.”. Esto nos permite verificar que la pieza artística se acople a lo del presente informe.

La última y la más importante, es la ejecución del diseño. En esta etapa es tomar la idea que mejor funcione y comprometerse con ella (Tempone, 2020).

Se eligió realizar un audiovisual mostrando la interacción del nuevo profesional ingresante y el buddy asignado. Dado el análisis que se realizó previamente se sabía de antemano que se iban a recurrir a cuatro situaciones en las cuáles había que demostrar en qué preguntas podría fallar el buddy, de esta manera se podía determinar en cuál de todas las situaciones el buddy era más eficiente y eficaz. Al mismo tiempo, los análisis de las respuestas que hubo, permiten demostrar esa transferencia de

conocimiento como el principal beneficio que puede haber si hay una persona con experiencia en el mismo rol.

El audiovisual se basó en seguir una serie de preguntas, entre las cuáles fueron las siguientes, aunque no todas fueron incluidas a fin de mantener el video acotado y que abarque quince minutos como máximo.

- ¿Qué tecnologías se manejan en el equipo?
- ¿Qué es Success Factors?
- ¿Qué herramientas se usan para la comunicación interna?
- ¿Qué repositorio se usa en el equipo?
- ¿El circuit braker es un patrón de diseño?
- ¿Cómo es el tema de la escalabilidad horizontal y vertical que se utiliza?
- ¿Qué patrones de diseño se utilizan en el equipo?
- ¿Cuál es la diferencia entre guardia pasiva y guardia activa?
- ¿Por qué se utiliza Datadog y New Relic? ¿Y Kibana?
- ¿Hay algún equipo que administre la base de datos?

Una vez que se realizó el video respondiendo a esas preguntas. Se utilizó Adobe Premiere para realizar la edición de a partes. Se fueron acomodando las preguntas en base a las cuatro situaciones; diferente equipo y diferente rol, diferente equipo y mismo rol, mismo equipo y diferente rol, mismo equipo y mismo rol. A su vez se agregaron placas que identifiquen la situación con las preguntas que se van a preguntar. Así como también se agregaron efectos de sonido con las transiciones en algunos casos. Por último, se agregaron unas pocas imágenes donde era posible incorporarlas, para demostrar la herramienta tecnológica de la que se estaba hablando. Por último, se agregó música, imágenes y voz en off.

En el siguiente link se puede ver el audiovisual[5]. Mientras que en el siguiente link se puede ver el audiovisual corto[6] para la defensa de la presente tesis.

[5] https://drive.google.com/file/d/1oJXFFig5xwnx3qKhHmSrL--y1wHK984R/view

[6] https://drive.google.com/file/d/1VWf5onKPw3TGyuQe8K_Sdj4GCHeDvlB_/view

Conclusiones

En base a todo lo expuesto durante el presente informe, podemos concluir que la hipótesis que se dispuso experimentar fue realizada con éxito, demostrando que la mejor forma de asignar un buddy es que sea del equipo perteneciente al nuevo profesional y que posee el mismo rol. Esto aumentará la productividad, el desarrollo mutuo entre ambas personas emparejadas, resultando en un amplio beneficio para el equipo. A su vez, quien tuvo el buddy asignado tiene la experiencia propia de haber pasado por esa situación, desencadenando en que puede convertirse en uno para otra persona.

El análisis previo tuvo como punto de partida dos experiencias diferentes. Lo que me permitió preguntarme para qué sirve y cómo tiene que ser un buen buddy. Pudiendo realizar el experimento con mis colegas dentro de la empresa, las respuestas no sólo me permitieron ver que cada uno que tuvo un buddy pasó una experiencia gratificante, sino que aquellos que no tuvieron vieron la necesidad de tener uno.

Un tema a destacar es que, gracias a las preguntas libres del formulario, una de las respuestas me demostró que una persona tuvo 2 buddies. Esto favoreció los resultados obtenidos debido a que en las fuentes de información nunca se habla sobre cuántos buddies pueden asignarse a una persona o si un buddy puede tener más de una persona a cargo.

La pieza de diseño es un audiovisual que no se tuvo en consideración ni el seniority de la persona que hace de buddy, ni tampoco su experiencia como desarrollador, ni la experiencia dentro de la empresa. La principal crítica que se le puede asignar es no haberse realizado con diferentes personas que cumplieran con las cuatro situaciones para realizar un experimento más realista. Sin embargo, las respuestas que me brindó la persona que actúa como buddy están un poco más profundizadas ya que posee una experiencia de 3 años dentro de la empresa y las preguntas fueron más generalizadas.

Adicionalmente hubiera sido interesante mostrar cómo es la relación de dos profesionales de desarrollo mostrando esa interacción a la hora de programar. Es decir, mostrar un poco de pair-programming con el buddy ya

que a veces requiere algo más puntual en el área de desarrollo como un vistazo por las carpetas y cómo es todo lo que se emplea en el código fuente. En una parte dada del audiovisual, se nombra el circuit breaker como patrón de diseño y hubiera sido interesante verlo desde el punto de vista del código, pero no es posible compartir código alguno y requiere tiempo para poder realizar un proyecto en el cual esté codeado y ambas personas estén al tanto de lo mismo como para realizar una demostración.

A su vez hay que destacar que, si bien las preguntas fueron generales, la experiencia de Leandro Devoto permitió una respuesta muy completa en cuanto al área de backend permitiendo también la transferencia de conocimiento para todo aquel que la escuche.

Finalizando con el presente informe, se agregan algunos detalles a tener en cuenta para todo aquel que quiera realizar el experimento a futuro. A su vez, estas consideraciones pueden favorecer la presente investigación.

Una posible variable a analizar puede ser con respecto al seniority de la unidad de análisis. Esto permitiría discriminar, si además de que el buddy sea del mismo equipo y del mismo rol, en si es necesario que el seniority sea del mismo ingresante o superior.

Otra posible variable para profundizar la hipótesis, tiene que ver con respecto a no limitar con saber si tuvo un buddy solo dentro de la empresa del caso de uso. Preguntándole a la persona si alguna vez tuvo un buddy en MercadoLibre u otra empresa, permitiría tener en consideración las respuestas libres del formulario empleado para la obtención de datos.

De acuerdo al formulario mostrado para la recolección de datos, una mejora puede realizarse con respecto a identificar el rol de las personas. Se propone cambiar el texto libre por una lista de selección, agregando como última opción otros, y que la misma persona decida agregar su cargo. Esto ayudaría en el análisis ya que se unificarán algunos cargos y se generaría una estadística automática.

Referencias

Abritta, Guillermo Pablo. (2014). Noción y estructura del dato [Archivo PDF]. Recuperado el 04 de julio de 2022 de http://metodos-comunicacion.sociales.uba.ar/wp-content/uploads/sites/219/2014/09/Abritta.pdf

Atlassian. Understanding Modern Incident Management with Opsgenie [Entendiendo la administracion de incidentes modernos con Opsgenie]. Recuperado el 04 de julio de 2022 de https://www.atlassian.com/software/opsgenie/what-is-opsgenie

Atlassian. ¿Para qué sirve Jira? Recuperado el 04 de julio de 2022 de https://www.atlassian.com/es/software/jira/guides/use-cases/what-is-jira-used-for#jira-for-agile-teams

Aulab. Las principales herramientas de un programador Web. Recuperado el 04 de julio de 2022 de https://aulab.es/noticia/76/las-principales-herramientas-de-un-programador-web

AWS. What are Microservices? [¿Qué son los microservicios?]. Recuperado el 04 de julio de 2022 de https://aws.amazon.com/microservices/

Ayushharwani2011. (2 de noviembre 2020). Pair Programming [Pair-programming]. Recuperado el 04 de julio de 2022 de https://www.geeksforgeeks.org/pair-programming/

Baeldung. Compiled vs. Interpreted Programming Languages [Lenguaje de programación compilados vs interpretados]. Recuperado el 04 de julio de 2022 de https://www.baeldung.com/cs/compiled-vs-interpreted-languages

Blancarte, Oscar. (7 de marzo de 2017). Escalabilidad Horizontal y Vertical. Oscar Blancarte Blog. Recuperado el 04 de julio de 2022 de https://www.oscarblancarteblog.com/2017/03/07/escalabilidad-horizontal-y-vertical/

Choudhary, Sneh Ratna. (09 de diciembre de 2021). How and Why to Create an Onboarding Buddy Program [Cómo y Porqué crear un programa inductive de buddy]. Zavvy. Recuperado el 04 de julio de 2022 de https://www.zavvy.io/blog/onboarding-buddy-program

Codeacademy. What Is an IDE? [¿Qué es un IDE?]. Recuperado el 04 de julio de 2022 de https://www.codecademy.com/article/what-is-an-ide

Codementor. Pair Programming: What, Why, and How [Pair-programming: Qué, Por qué, Cómo]. Recuperado el 04 de julio de 2022 de https://www.codementor.io/pair-programming

COR. Roles Fundamentales en un Equipo de Desarrollo de Software. Recuperado el 04 de julio de 2022 de https://projectcor.com/es/blog/roles-fundamentales-en-un-equipo-de-desarrollo-de-software/

Evan Leybourn. (27 de junio 2013). Directing The Agile Organization: A Lean Approach To Business Management [Dirigiendo una organización ágil: Un enfoque inclinado hacia la administración empresarial]. Recuperado el 04 de julio de 2022 de https://theagiledirector.com/images/IntroductiontoScrum-coursenotes.pdf

Guthals, Sarah & Haack, Phil. (2019). GitHub For Dummies [Github para tontos]. John Wiley & Sons, Inc.

Hamilton, Thomas. (30 de abril de 2022). What is Software Testing? Definition, Basics & Types in Software Engineering [¿Qué es prueba de software? Definición, principios basicos y tipos en ingeniería de software]. Recuperado el 04 de julio de 2022 de https://www.guru99.com/software-testing-introduction-importance.html

Hooda, Parikshit. (6 de agosto de 2021). Introduction to Postman for API Development [Introducción a Postman para desarrollar una API]. Recuperado el 04 de julio de 2022 de https://www.geeksforgeeks.org/introduction-postman-api-development/

humanfusion. (09 de junio de 2019). Every New Employee Needs an Onboarding "Buddy" (Part 1) [Archivo de Video]. Youtube. Recuperado el 04 de julio de 2022 de https://www.youtube.com/watch?v=1TANfL5hic8

Indeed. Creating a Buddy System in the Workplace [Creando un Sistema de Buddy en el trabajo]. Recuperado el 04 de julio de 2022 de https://www.indeed.com/hire/c/info/buddy-system

Indeed Editorial Team. (23 de diciembre de 2020). Project Leader vs. Project Manager: Definitions and Key Differences. Indeed. Recuperado el 04 de julio de 2022 de https://www.indeed.com/career-advice/finding-a-job/project-leader-vs-project-manager

JavaTpoint. Database [Base de datos]. Recuperado el 04 de julio de 2022 de https://www.javatpoint.com/what-is-database

Justin. (11 de enero de 2022). What does New Relic do? [¿Qué hace New Relic?]. Technically**.** Recuperado el 04 de julio de 2022 de https://technically.substack.com/p/what-does-newrelic-do?s=r

Lindley, Cody. (2019). Front-end Developer Handbook 2019 [manual de desarrollador frontend 2019]. Frontend Masters.

MercadoLibre. Ecosistema Mercado Libre: el valor de pensar todas las soluciones. Recuperado el 04 de julio de 2022 de https://www.mercadolibre.com.ar/institucional/hacemos/ecosistema-mercado-libre

MongoDB. Relational vs. Non-Relational Databases [Base de datos relacionales vs no relacionales]. Recuperado el 04 de julio de 2022 de https://www.mongodb.com/compare/relational-vs-non-relational-databases

Mullins, Laurie J. (2005). Management and Organisational behaviour [Administración y comportamiento organizacional]. Pearson Education Limited.

Myers, Dominic. (2020). FRONT-END DEVELOPER [Desarrollador frontend]. Learning & Development Ltd.

Osiri, David. (2020). PRACTICAL STEPS TO FINDING A MENTOR [Pasos prácticos para encontrar un mentor]. Independently.

Sagar. (12 de mayo de 2022). What is Datadog – The Ultimate Guide [Qué es Datadog – La última guía]. Petri. Recuperado el 04 de julio de 2022 de https://petri.com/what-is-datadog/

QuestionPro. Cuáles son los tipos de variables en una investigación. Recuperado el 04 de julio de 2022 de https://www.questionpro.com/blog/es/tipos-de-variables-en-una-investigacion/

Ramotion (02 de junio de 2021). Ultimate Back End Developer Guide [La ultima guía de desarrollador Backend]. Recuperado el 04 de julio de 2022 de https://www.ramotion.com/blog/back-end-developer-guide/

RevDeBug. (15 de abril de 2021). Unit Tests Vs. Integration Tests [Pruebas unitarias vs pruebas de integración]. Recuperado el 04 de julio de 2022 de https://revdebug.com/blog/unit-tests-vs-integration-tests/

Roy, Sandip. 1 de enero de 2022. The Difference Between a Framework and a Library [La diferencia entre un framework y una librería]. Recuperado el 04 de julio de 2022 de https://www.baeldung.com/cs/framework-vs-library

Sánchez, C. (08 de febrero de 2019). Títulos y Subtítulos. Normas APA (7ma edición). Recuperado el 04 de julio de 2022 de https://normas-apa.org/formato/titulos-y-subtitulos/

Sánchez, C. (08 de febrero de 2019). ¿Cómo citar una fuente de un idioma extranjero y su traducción? Normas APA (7ma edición). Recuperado el 04 de julio de 2022 de https://normas-apa.org/citas/como-citar-una-fuente-de-un-idioma-extranjero-y-su-traduccion/

Sánchez, C. (24 de enero de 2020). Referencias APA. Normas APA (7ma edición). Recuperado el 04 de julio de 2022 de https://normas-apa.org/referencias/

Sánchez, C. (05 de mayo de 2020). Citar un Blog – Referencias Bibliográficas. Normas APA (7ma edición). Recuperado el 04 de julio de 2022 de https://normas-apa.org/referencias/citar-un-blog/

Sánchez, C. (29 de enero de 2020). Tablas. Normas APA (7ma edición). Recuperado el 04 de julio de 2022 de https://normas-apa.org/estructura/tablas/

Saavedra R., Manuel S. (2001). Elaboración de tesis profesionales. Pax México.

Slack. ¿Qué es Slack? Recuperado el 04 de julio de 2022 de https://slack.com/intl/es-ar/help/articles/115004071768-%C2%BFQu%C3%A9-es-Slack-

SourceMaking. Design Patterns [Patrones de diseño]. Recuperado el 04 de julio de 2022 de https://sourcemaking.com/design_patterns#:~:text=In%20software%2

0engineering%2C%20a%20design,used%20in%20many%20different%20situations.

Sumo logic. Software Deployment [Deployment de software]. Recuperado el 04 de julio de 2022 de https://www.sumologic.com/glossary/software-deployment/

Swagger. What Is Swagger? [¿Qué es Swagger?]. Recuperado el 04 de julio de 2022 de https://swagger.io/docs/specification/2-0/what-is-swagger/

Tempone, Denise. (05 de enero de 2022). ¿Qué es el proceso de diseño y cuales son los diferentes pasos? Domestika. Recuperado el 04 de julio de 2022 de https://www.domestika.org/es/blog/9704-que-es-el-proceso-de-diseno-y-cuales-son-los-diferentes-pasos

Wozniewicz, Brandon. 1 de febrero de 2019. The Difference Between a Framework and a Library [La diferencia entre un framework y una librería]. Recuperado el 04 de julio de 2022 de https://www.freecodecamp.org/news/the-difference-between-a-framework-and-a-library-bd133054023f/