Water Data Project para Groots

Architectura - Definición por FASES

[**Alcance**](#_yv6byj9q7wba) **2**

[**Requerimientos**](#_z9mp7zkiqqju) **2**

[Requerimientos Funcionales](#_th2v4iaprqys) 2

[Requerimientos No Funcionales](#_u42elpb81868) 3

[Características Técnicas](#_6d22d0mdfhl0) 3

[**Requisitos de Nexion**](#_vpwcenwgu2bp) **3**

[Requisitos Funcionales](#_woiqxd66udfs) 3

[Requisitos Tecnológicos](#_6lprwgd0884v) 4

[**Comprensión de los requerimientos**](#_7hjhaam5l81c) **4**

[Arquitectura FASE 1](#_3u7g7ttf8b91) 5

[Principales Características](#_5gxuzn5cnbwf) 5

[Consideraciones FASE 1](#_u14bs8bgk2wv) 6

[Arquitecturas Candidata FASE 2](#_8pwwy9u1yxvm) 7

[Arquitectura Candidata A](#_284d4p4hxzod) 8

[Principales Características - Candidata A](#_7h8s24bmret) 8

[Componentes](#_brq6n32mtiql) 9

[Support API](#_6hty4j2id2rl) 9

[KoBoCollect / ODKCollect](#_nb0pa4f3jrln) 10

[KoBoToolbox Services](#_5f40671yejfq) 11

[Capas de la Aplicación](#_dgfx4fgi7x1s) 12

[Diagrama de contenedores](#_s9q9lp1tzthk) 13

[Diagrama de Despliegue](#_oi3mmsartj5u) 14

[Flujo de Desarrollo](#_nyqtcqv90liq) 15

[Consideraciones Finales](#_j2pwr7jg8rc5) 16

[Beneficios](#_7fiaif4ucx) 16

[Skills requeridos](#_xn9tysa5wrbw) 16

[Riesgos](#_pwjdrbixe0rl) 16

[Veredicto](#_ck8d76yg92bw) 17

[Arquitectura Candidata B](#_j5030tj0swot) 17

[Principales Características - Candidata B](#_1cp9r9d70i6r) 17

[Componentes](#_9mb8mqd7l8au) 19

[Support API](#_km1t4tgjwl58) 19

[Ver a definición en Candidata A](#_4yciyb3uybjt) 19

[KoBoCollect / ODKCollect](#_kfs24nlp38t) 19

[KoBoToolbox Services](#_kumsnxlm6v1e) 20

[Capas de la Aplicación](#_rl51bsm0dbat) 21

[Diagrama de contenedores](#_3gxi0iqj1i1c) 22

[Diagrama de Despliegue](#_q66h49dge4et) 23

[Flujo de Desarrollo](#_acw9t14dpy0z) 24

[Consideraciones Finales](#_4blo55cxn9ij) 25

[Beneficios](#_jc95oawzvlhp) 25

[Skills requeridos](#_wvlfw9qs1abc) 25

[Riesgos](#_a6wpqvzf9n0h) 25

[Veredicto](#_43ad20q7kx62) 26

[Arquitectura Candidata B](#_nrhd15okgzde) 26

# Alcance

Realizar el análisis de la arquitectura a desarrollar y desplegar para su futuro mantenimiento y/o escalado.

Se proponen dos fases de trabajo:

* **FASE 1**: Instalación de KoBoToolbox. Operativa completamente basada en la funcionalidad de KoBoToolbox.
* **FASE 2**: Implementación de Incentivos en una aplicación web complementaria a KoBo. La misma tendría la capacidad de manejar Incentivos y basarse en Role Management para un acceso granular.

Se presenta una arquitectura para la **FASE 1**

Se presentan dos arquitecturas candidatas en base a la comprensión de los requerimientos de **FASE 2**

Ambas fases serán indicadas a lo largo del documento

Se diseña la arquitectura en base a la comprensión de la herramienta de gestión y levantamiento de encuestas: **KoBoToolbox**.

La arquitectura candidata debe ser validada con Pruebas de Concepto (POC por sus siglas en inglés).

Este documento es considerado de alto nivel y no es una especificación técnica de implementación por lo que las especificaciones de implementación serán detalladas en documentos complementarios y desarrolladas por el equipo.

# Requerimientos

## Requerimientos Funcionales

**FASE 1**

* Contar con una herramienta de levantamiento de encuestas
  + Capacidad de generar nuevas encuestas.
  + Capacidad de editar y gestionar encuestas existentes.
  + Capacidad de levantar encuestas en poblados alejados sin conexión a internet.
  + Capacidad de analizar los datos enviados por los encuestadores de manera de poder hacer el análisis de datos que soporte a la toma de decisiones.

**FASE 2**

* Contar con una herramienta de incentivos
  + Poder enviar incentivos a los Comités de Agua Potable y Saneamiento a través de una aplicación.
  + [A ser delimitado]. **No se cuenta con mayores especificaciones.**
* Contar con niveles de acceso a diferentes lugares de la herramienta

## Requerimientos No Funcionales

* La herramienta debe ser de naturaleza OpenSource.
* La herramienta debe ser desplegada en un entorno OpenSource.
* El acceso a internet no está garantizado, por lo que las herramientas a plantear deben ser sincronizables cuando internet esté disponible (modo offline).
* Las herramientas de Gestión (Definición y análisis de datos) no necesitan ser de modo Offline.

## Características Técnicas

**FASE 1**

* Los servicios a ser entregados deben ser instalados en la infraestructura local de la entidad beneficiaria de la herramienta.
* Para bajar costos de desarrollo, se sugiere evaluar herramientas de levantamiento de encuestas:
  + KoBoToolbox u
  + ODK Central

**FASE 2**

* La herramienta de incentivos puede ser OpenSource pero también es posible evaluar una herramienta a medida, con elementos básicos, una vez que el alcance de la misma haya sido definido por Groots.
* Es deseable que el esquema de autenticación sea de tipo Single-SignOn. Por lo que la evaluación de las herramientas deberían proveer un estándar de autenticación abierto como OAuth2.0
  + En caso de no contar con esta característica, el proyecto sigue siendo viable y el equipo de Nexion debe elaborar documentación para el correcto uso de aplicaciones separadas.

# Requisitos de Nexion

## Requisitos Funcionales

El principal objetivo es que las encuestas se realicen en poblaciones con un limitado o incluso inexistente acceso a la red internet.

* Los usuarios de la aplicación de levantamiento de datos deben contar con una herramienta simple
  + Es posible que sea necesaria la capacitación por parte de personal de Groots/Gobernación/otros
* La herramienta debe estar disponible en español
* Los encuestadores deben saber leer en español y manejar un celular con sistema operativo Android
* Los administradores de la herramienta deben tener los siguientes conocimientos:
  + Uso de aplicaciones basadas en Web y Móvil (Android)
  + Descarga de datos para análisis en Excel, Google Sheets u otra herramienta
* El módulo de Incentivos debe ser delimitado y especificado para la puesta en marcha de la **FASE 2**

## Requisitos Tecnológicos

* Los encuestadores deben contar con un teléfono Android con una versión mínima que es soportada por la aplicación de levantamiento de encuestas (KoBoCollect o ODKCollect)
* El personal técnico de la entidad beneficiaria debe tener los siguientes conocimientos para gestionar el uptime de la aplicación:
  + Gestión de servidores Linux
    - Instalación, mantenimiento
    - Gestión de redes (DNS, publicación de una página web al exterior)
  + Gestión de contenedores con Docker
    - Nivel básico a intermedio
    - Lectura de logs
  + Certificados SSL
    - Instalación, mantenimiento
  + Lenguajes de programación para despliegue
    - Básico de python, yml, dotnet core, bash
    - Conocimiento de Git
* (Opcional) El equipo de desarrollo debe contar con las siguientes aptitudes

*Si los roles de los beneficiarios no son claros respecto al ciclo de vida del software se recomienda no realizar actividades adicionales de desarrollo*

* + Plataformas
    - Linux (Basado en Debian o Fedora)
    - Docker
  + Lenguajes de desarrollo / scripting
    - Javascript ES6
    - Python 3
    - Dotnet core 3.1
    - React
    - (Por Definir) Flutter
    - Bash
  + IDE
    - Visual Studio Code / (opcional) Visual Studio Community
    - (opcional) Pycharm
  + Otros
    - Git

# Comprensión de los requerimientos

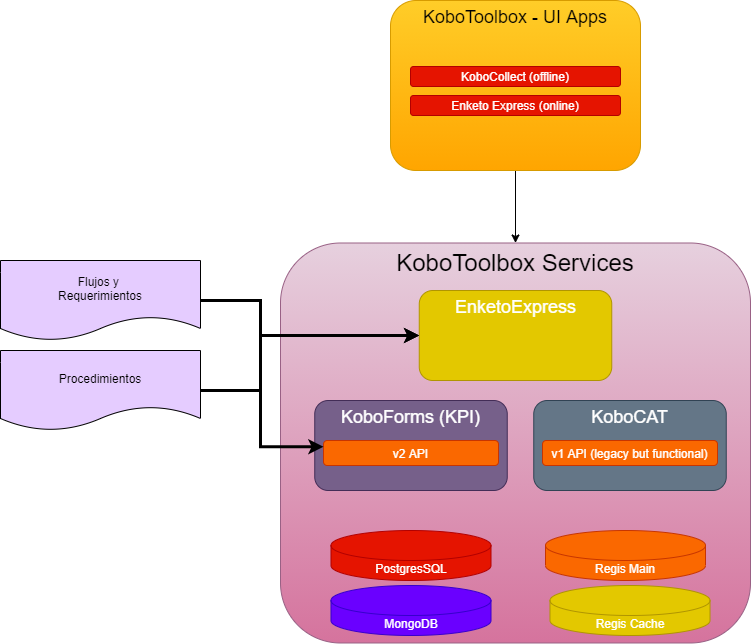
**FASE 1**

En base a lo expuesto en los puntos anteriores se tienen los siguientes criterios técnicos de cara a la implementación:

* El sistema de levantamiento de encuestas debe ser un producto OpenSource ya existente. Por lo que el desarrollo debe montarse en torno a éste a través de API
* En base al documento: “***Architectura - KoBoToolbox vs ODK - Pros y Cons***” El sistema de levantamiento de encuestas elegido es **KoBoToolbox.** Los elementos comparativos se pueden encontrar en la sección: **KoBoToolbox vs ODK - Por criterio.** La decisión en base a un puntaje obtenido se pueden encontrar en la sección: **Puntuación**

A continuación se hará un desglose de los elementos arquitecturales que satisfacen los puntos mencionados anteriormente.

## Arquitectura FASE 1



#### Principales Características

* La **FASE 1** se concentra completamente en la puesta en marcha de KoBoToolbox en los servidores de la contraparte beneficiaria.
* El principal objetivo es generar la puesta en marcha de la aplicación de levantamiento de formularios lo antes posible para manejar KoBoToolbox independientemente. Esto es importante para generar aprendizaje validado en campo de manera de poder contar con datos que justifiquen la puesta en marcha de este proyecto.
* Al momento de presentar este documento, los requerimientos funcionales de la **FASE 2** no han sido totalmente especificados, ya que la fase de evaluación en campo aún se encuentra en proceso. Esto es fundamental para dar una solución más cercana del lado de una plataforma de software, por lo que el éxito de la **FASE 1** permite que **FASE 2** se lleve a cabo ya con mayores inputs para la delimitación del proyecto.
* **FASE 1** se trata completamente de operativizar la generación de encuestas y levantamiento en campo. Esto también significa conocer los roles de los usuarios/instituciones/CAPYS/etc en cuanto a la necesidad de consumo de información
* **FASE 1** tiene el objetivo de generar aprendizaje, validado y contrastado a través de el pronto uso de la herramienta en campo
* Se debe realizar un manual operativo inicial de uso de la herramienta en base a los supuestos actuales

#### Consideraciones FASE 1

* Restricciones de tiempo y presupuesto
* El tiempo para una implementación está medido a pocas semanas. Esto justifica la implementación inmediata de KoBoToolbox sin customizaciones, como una caja negra
* **Si el presupuesto no permite crear una aplicación complementaria. La misma permitiría la importación de datos de encuestas para ser otorgadas a través de roles específicos hacia entidades y actores del proyecto**
* Riesgos técnicos
  + La herramienta KoBoToolbox es una caja negra. Querer modificar el comportamiento interno en favor de customizaciones específicas al negocio/proyecto, requiere de un conocimiento técnico del lenguaje y tecnología que utiliza internamente, lo cual no garantiza que cualquier elemento modificado en su interior mantendría la integridad de la herramienta, es decir se pueden generar errores de programación (bugs) que deben ser resueltos por el equipo
  + Cualquier customización de KoBoToolbox debe ser responsabilidad del proyecto en cuanto a tiempos adicionales a incurrir
* El tiempo de **FASE 1**
  + Ha sido el tiempo de investigación y conocimiento de la herramienta a través de instalaciones de la misma en diversos ambientes para conocer las capacidades y requerimientos
  + Adicionalmente se deben realizar manuales de usuario, de flujos y procedimientos
* De la instalación
  + **FASE 1** **no** considera la instalación de la herramienta dentro de los servidores del beneficiario del proyecto. Hasta el momento de la elaboración de este proyecto, no se tiene definido al beneficiario específico que recibiría la responsabilidad de alojar al sistema en sus instalaciones.
  + Se sugiere contar con el personal receptor de la herramienta para su mantenimiento.

## Arquitecturas Candidata FASE 2

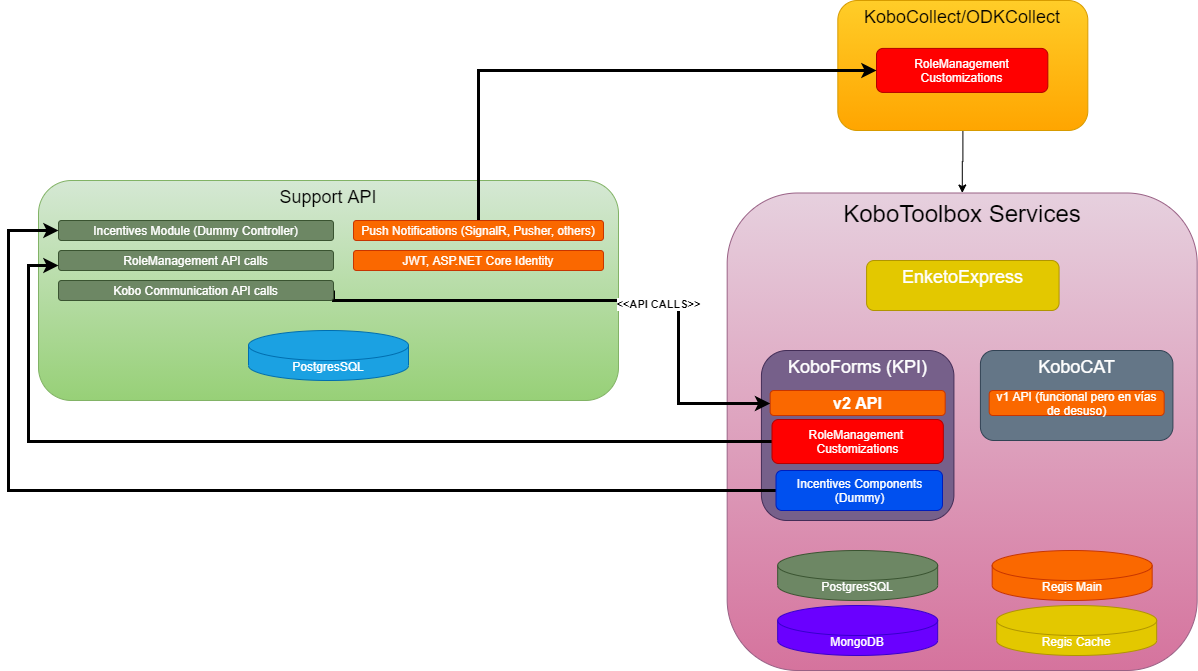
Cada arquitectura candidata muestra los elementos técnico-funcionales desde una perspectiva de componentes e interrelación.

Se presentarán dos arquitecturas:

* **Arquitectura Candidata A**
  + Customización de ODKCollect/KoBoCollect.
  + Customización de KoBoForms(KPI) para adaptarse a las necesidades de los requerimientos de Incentivos insertando un módulo desarrollado por Nexion que corra dentro de KoBoForms.
  + 1 Servicio con todas las APIs.
  + 1 Base de datos en total, independiente al server de KoBoToolbox.
  + Se modifica el código de la app de ODKCollect o KoBoCollect para incluir lógica de RoleManagement para el uso de Incentivos
  + Se modifica el código de la app KoBoForms.
* **Arquitectura Candidata B**
  + Customización de ODKCollect /KoBoCollect.
  + **Custom App de Incentivos.**
  + 1 Servicio con todas las APIs.
  + 1 Base de datos en total.
  + Se modifica el código de la app de ODKCollect o KoBoCollect para incluir lógica de RoleManagement para el uso de Incentivos.
  + Se modifica el código de la app KoBoForms.

|  |
| --- |
| ***Importante: Se ha desestimado la inclusión de conceptos basados en microservicios: Gateway y Message Broker debido a que este proceso haría más complejo el desarrollo y está pensado para crecer en muchos dominios de negocio con equipos de desarrollo, hecho que no se considera para implementación actual dados los requerimientos de tiempo y presupuesto.*** |

### Arquitectura Candidata A

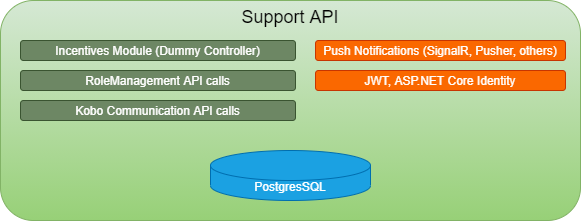


#### Principales Características - Candidata A

* Se realizan customizaciones de **KoBoToolbox** para adaptarse a definiciones de RoleManagement en función de implementar Incentivos
  + Significa que existirían llamadas a la API de RoleManagement en *Support API.*
    - Esto requiere que se modifique la estructura de deployment en Docker de KoBoToolbox, específicamente agregar la nueva API (*Support API*) como parte del paquete Docker (docker-compose).
    - El código a modificar debe ser parte de un FORK (en github) de la versión de KoBoToolbox, por lo que se tendría una versión independiente a nuevos releases de KoBoToolbox.
    - **La estabilidad de la aplicación KoBoToolbox no está garantizada por lo que se puede incurrir en tiempos adicionales ante una posible necesidad de estabilización.**
* Se realizan cambios en ODKCollect/KoBoCollect. Éstos deben ser retro-compatibles con KPI, KoBoCAT y Enketo Express.
  + Significa que cualquier cambio debe seguir usando la API v2 de KPI.
  + Requiere acceso a los servicios API de *Support API.*
  + El código a modificar debe ser parte de un FORK (en github) de la versión de actual de la app de ODKCollect/KoBoCollect, por lo que se tendría una versión independiente a nuevos releases.
  + Se debe pensar en la publicación de esta aplicación en Google PlayStore o hacerla disponible a través de otra plataforma.
* También se tiene acceso a KoBoToolbox es a través de la API v2 que es expuesta por KoBo Forms (KPI) utilizando API Keys (disponibles por aplicación web y llamada REST).
  + Se sugiere tener un sólo usuario (con capacidad IsAdmin) que realice todas las llamadas al API de KoBoToolbox.
* Respecto al **Backend**
  + Todos los servicios se crean bajo el mismo servicio monolítico.
  + Servicios:
    - API de RoleManagement.
    - API de Incentivos (dummy controller hasta su definición formal)
  + Una sola base de datos: Postgresql
    - Se considera que por simplicidad, se puede usar la instancia de PostgreSQL de KoBoToolbox (container) si el modelo de deployment no resulta demasiado complejo de modificar. La misma que es una implementación de PostGIS.
    - Alternativamente. Se crea una base de datos totalmente independiente a los usados por KoBoToolbox, sin embargo el DBMS (Server) es el mismo.
      * Se evaluará al momento de realizar la implementación si es justificado crear un nuevo container de PostgreSQL si el esfuerzo de modificar los scripts de KoBoToolbox no se justifica.
  + El módulo de incentivos necesita la funcionalidad de push notifications.
    - La tecnología puede ser SignalR, Pusher u otro gestor de websockets.
  + El módulo de RoleManagement puede hacer uso de ASP.NET Core Identity
    - No se recomienda construir una customización de Role Management ya que se busca una funcionalidad básica de roles, usuarios y gestión de los mismos.
    - Se requiere que la API implemente OAuth2.0, por lo que se requiere usar tokens tipo JWT.
* Respecto al **Frontend**:
  + **Customización de ODKCollect/KoBoCollect**
    - Se insertan elementos de Flutter para poder realizar ajustes adaptados a las necesidades del módulo de incentivos.
    - Puede hacer comunicación a la API de Support API.
    - Se deberá publicar la aplicación en Google PlayStore como una aplicación independiente a ODKCollect/KoBoCollect.

#### Componentes

##### Support API



|  |  |
| --- | --- |
| **Concepto** | **Valor** |
| Objetivo | Crear una plataforma API basada en dotnet core para ser parte de los servicios a ser desplegados en el entorno de despliegue de KoBoToolbox. |
| Plataforma API | Basada en una API de tipo ASP.NET Core 3.1/5 |
| Módulos | * Incentivos * Role Management * Comunicación con KoBo |
| Base de datos | PostgresSQL |
| Componentes de soporte | * Push Notifications   + SignalR   + Pusher * ASP.NET Core Identity * JWT |

[También en Candidata B](#_km1t4tgjwl58)

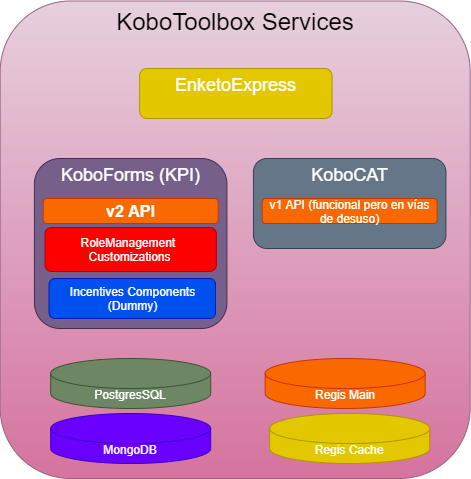
##### KoBoCollect / ODKCollect



|  |  |
| --- | --- |
| **Concepto** | **Valor** |
| Objetivo | Customización de KoBoCollect / ODKCollect para incrustar lógica específica al proyecto. |
| Plataforma Móvil | * Android * Flutter |
| Módulos | * Role Management * Incentivos |
| Componentes Flutter | Se deben insertar en el proyecto Android |

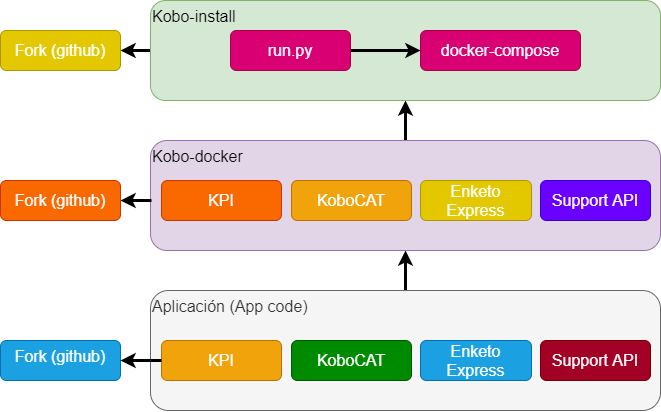
[También en Candidata B](#_kfs24nlp38t)

#### KoBoToolbox Services



|  |  |
| --- | --- |
| **Concepto** | **Valor** |
| Objetivo | Customización de KPI para soportar implementaciones específicas de Incentivos y acceso por RoleManagement |
| Plataforma | React (Server side rendering) |
| Módulos | * Incentivos * RoleManagement (sólo para accesos) |

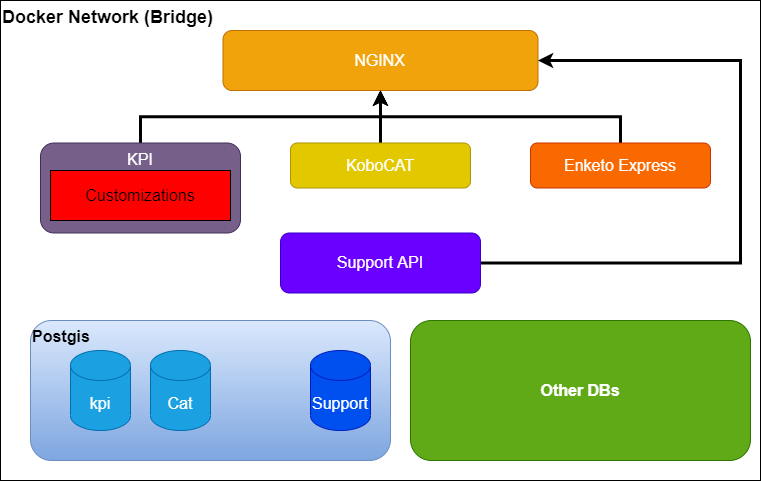
#### Capas de la Aplicación



|  |  |
| --- | --- |
| **Concepto** | **Valor** |
| Capa: Aplicación | * Se debe realizar un Fork del proyecto [KPI en Github](https://github.com/kobotoolbox/kpi) para realizar los cambios necesarios en el código |
| Capa: KoBo-docker | * Se debe hacer Fork del proyecto [KoBo-docker en Github](https://github.com/kobotoolbox/kobo-docker) para realizar los cambios en los scripts para las acciones docker:   + Build   + Tag   + Push * Se hará un push a DockerHub de la nueva imagen de KPI generada   + Este es el insumo para Docker-compose |
| Capa: KoBo-install | * Se debe hacer Fork del proyecto [KoBo-install en Github](https://github.com/kobotoolbox/kobo-install) * Se deben ajustar los valores de git checkout para apuntar al fork correcto de KoBo-docker |

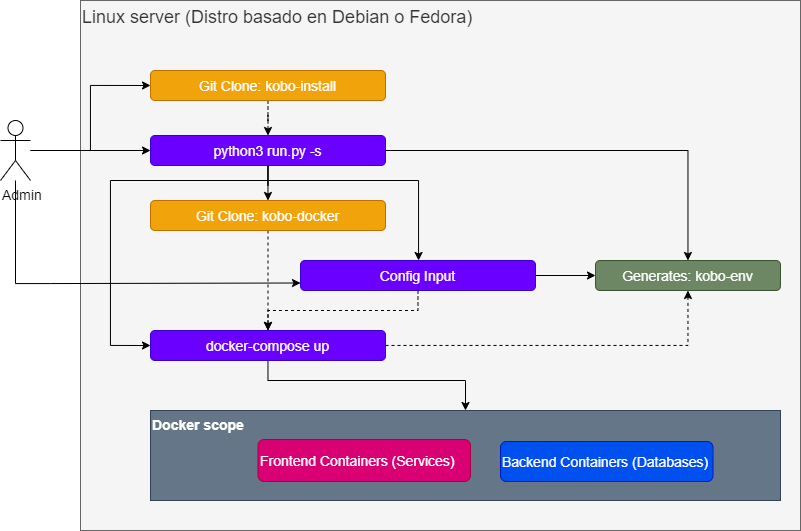
[También en Candidata B](#_uzywrcebx4a)

#### Diagrama de contenedores



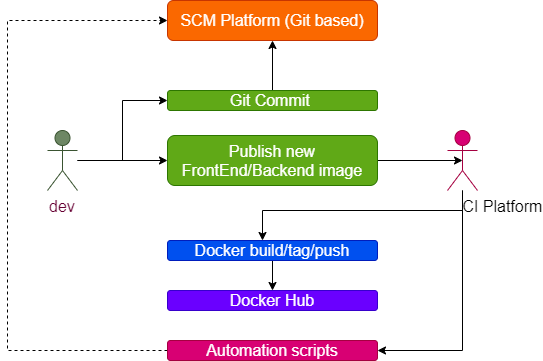
|  |  |
| --- | --- |
| **Concepto** | **Valor** |
| Contenedores de servicios | * KPI   + Este sería el contenedor con código customizado * KoBoCAT * Enketo Express * **Support API**. Éste contenedor sería generado por el proyecto |
| Base de datos | * Tipo: PostgresSQL * Esquema: Support * Locación:   + Sugerido: Dentro del server usado por KoBoToolbox   + Segunda opción: Tener un container propio basado en PostgresSQL 12 |
| Docker Compose | * Se debe agregar el servicio de Support API a los contenedores del mismo Network Bridge de docker-compose en el repositorio (fork) de KoBo-docker * Support API está bajo la misma red de KPI, KoBoCAT y Enketo Express * Los servicios anteriormente descritos deben poder acceder al servicio Support API |
| Healthcheck | * Support API debe publicar un endpoint healthcheck básico que retorne un status 200 y un mensaje de texto “HEALTHY” * Support API debe publicar un endpoint healthcheck de **base de datos** que retorne un status 200 y un mensaje de texto “DB HEALTHY” |

#### Diagrama de Despliegue



|  |  |
| --- | --- |
| **Concepto** | **Valor** |
| Operador | Admin de servidor Linux |
| Roles | * Sudoer * Linux Group: Docker * Admin del socket de Docker * Acceso al repo de GIT |
| Acciones del Admin | * Git Clone de KoBo-docker (Fork) * Ejecución de python3 run.py -s * Proveer valores de configuración * Verifica que todos los servicios estén activos y funcionales a través de:   + docker ps   + docker logs   + python3 run.py -l   + python3 run.py -i   + docker exec -it <container> bash |
| Acciones de run.py | * Realiza git clone de KoBo-docker * Crea el directorio KoBo-env en base a los valores de configuración provistos por el Admin * Ejecuta las acciones de docker compose para crear (o recrear) el ambiente |

#### Flujo de Desarrollo



|  |  |
| --- | --- |
| **Concepto** | **Valor** |
| Actores | * Dev   + Developer del proyecto * Plataforma CI   + Plataforma de integración contínua   + ej. Jenkins, GItLab CICD, AzureDevOps, etc |
| Acciones Dev | * Git Commit de cambios de código * Solicitar a la Plataforma CI que genere un nueva imagen docker del servicio desarrollado |
| Acciones Plataforma CI | * Gestionar scripts de automatización * Ejecución de scripts docker. Acciones:   + build   + tag   + push * Gestión de repositorio Git para obtener código a ser convertido en docker images |

[También en Candidata B](#_acw9t14dpy0z)

#### Consideraciones Finales

##### Beneficios

* No requiere una web app adicional
* No requiere una mobile app adicional
* Es posible extender ambas aplicaciones: KPI y KoBoCollect/ODKCollect con lógica específica

##### Skills requeridos

* Se requiere que el equipo de desarrollo cuente con conocimiento en la versión Android manejada por KoBoCollect/ODKCollect para realizar customizaciones
  + Conocimientos en publicación de aplicaciones Android en PlayStore
* Se requiere que el equipo de desarrollo cuente con conocimiento en la versión React (Server side render) utilizada en KPI (KoBoForms)
* Se requiere que el equipo de desarrollo cuente con conocimiento en Docker y Docker-compose para la modificación de los scripts

##### Riesgos

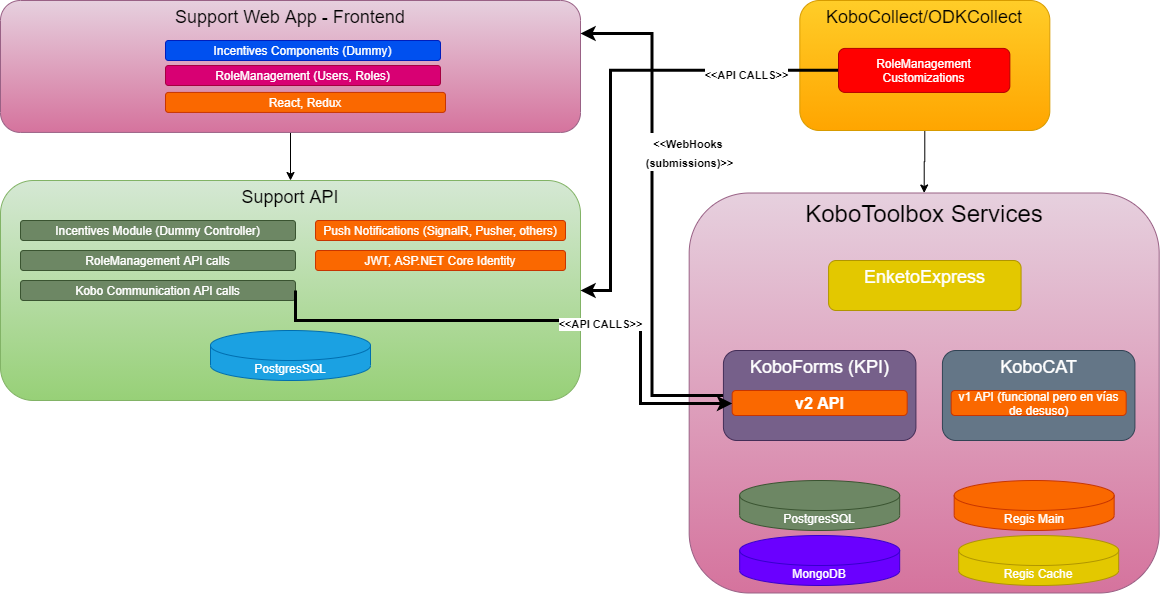
|  |  |
| --- | --- |
| **Tipo** | **Riesgo (Alto, Medio, Bajo)** |
| KoBoCollect/ODKCollect no permite que se customice la aplicación | Bajo |
| KoBoCollect/ODKCollect no puede ejecutar funciones específicas de su comportamiento | Bajo |
| KoBoCollect/ODKCollect podría quedar inestable luego de la customización | Medio |
| KPI tiene una curva de aprendizaje alta | Alto |
| KPI podría quedar inestable luego de la customización requiriendo más días de trabajo | Alto |
| Cualquier error de programación (bug) durante operaciones requiere de los servicios de Nexion para la corrección/estabilización | Medio |
| El trabajo de modificación de docker-compose y puesta en marcha requiere comprensión completa del repo KoBo-docker | Alto |

##### Veredicto

* Esta arquitectura es la que mejor se adapta a las necesidades de tiempo y presupuesto con las que se cuenta.
* Si se trabaja en enfocarse en minimizar los riesgos, ésta arquitectura es viable

### 

### Arquitectura Candidata B



#### Principales Características - Candidata B

* Se realizan cambios en ODKCollect/KoBoCollect. Éstos deben ser retro-compatibles con KPI, KoBoCAT y Enketo Express.
  + Significa que cualquier cambio debe seguir usando la API v2 de KPI.
  + Requiere acceso a los servicios API de *Support API.*
  + El código a modificar debe ser parte de un FORK (en github) de la versión de actual de la app de ODKCollect/KoBoCollect, por lo que se tendría una versión independiente a nuevos releases.
  + Se debe pensar en la publicación de esta aplicación en Google PlayStore o hacerla disponible a través de otra plataforma.
* También se tiene acceso a KoBoToolbox es a través de la API v2 que es expuesta por KoBo Forms (KPI) utilizando API Keys (disponibles por aplicación web y llamada REST).
  + Se sugiere tener un sólo usuario (con capacidad IsAdmin) que realice todas las llamadas al API de KoBoToolbox.
* Se usa la capacidad de notificación de eventos de KoBoToolbox (REST notifications) ante la necesidad de contar con eventos por envío de encuestas hacia *Support API.*
* Respecto al **Backend**
  + Todos los servicios se crean bajo el mismo servicio monolítico
  + Servicios:
    - API de Role Management.
    - Endpoint para capturar eventos de publicación de KoBoToolbox (REST Services).
    - API de Incentivos (dummy controller hasta su definición formal).
  + Una sola base de datos: Postgresql
    - Se considera que por simplicidad, se puede usar la instancia de PostgreSQL de KoBoToolbox (container).
    - Se crea un esquema totalmente independiente a los usados por KoBoToolbox
  + El módulo de incentivos necesita la funcionalidad de push notifications.
    - La tecnología puede ser SignalR, Pusher u otro gestor de websockets.
  + El módulo de RoleManagement puede hacer uso de ASP.NET Core Identity.
    - No se recomienda construir una customización de RoleManagement ya que se busca una funcionalidad básica de roles, usuarios y gestión de los mismos.
    - Se requiere que la API implemente OAuth2.0, por lo que se requiere usar tokens tipo JWT.
* Respecto al **Frontend**, se crean dos aplicaciones Frontend:
  + **Customización de ODKCollect/KoBoCollect.**
    - Se insertan elementos de Flutter para poder realizar ajustes adaptados a las necesidades del módulo de incentivos.
    - Puede hacer comunicación a la API de Support API.
    - Se deberá publicar la aplicación en Google PlayStore como una aplicación independiente a ODKCollect/KoBoCollect.
  + **Support Web App**
    - Destinada a la gestión de usuarios, Configuración de incentivos, Llamadas a la API v2 de KoBo Forms.
    - Plataforma web: React.

#### Componentes

##### Support API

La definición de esta arquitectura es igual a la arquitectura A

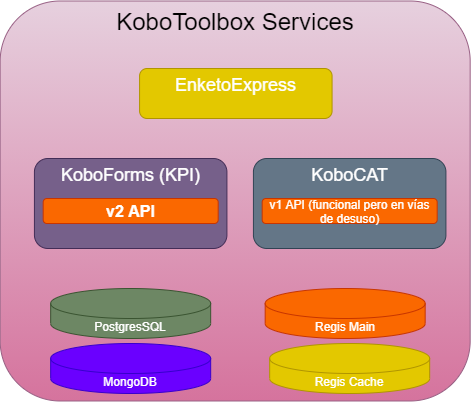
[Ver a definición en Candidata A](#_6hty4j2id2rl)

##### KoBoCollect / ODKCollect

La definición de esta arquitectura es igual a la arquitectura A

[Ver a definición en Candidata A](#_quwfob7ug3q5)

##### KoBoToolbox Services



|  |  |
| --- | --- |
| **Concepto** | **Valor** |
| Objetivo | Mantener la implementación de KoBoToolbox sin modificar. Todas las interacciones se realizan a través del API que KPI publica (v2) y a través de webhooks cuando se envían encuestas (si aplica) |
| Plataforma | React (Server side rendering) |

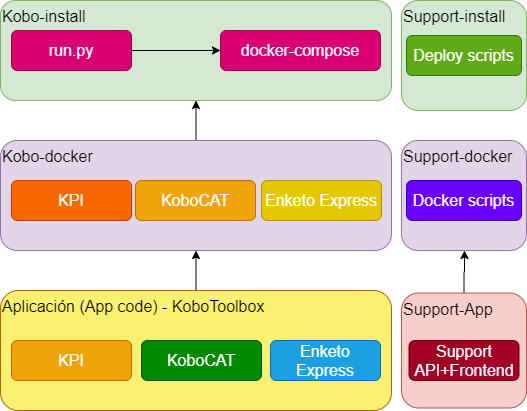
##### Support Web App - FrontEnd

#### 

|  |  |
| --- | --- |
| **Concepto** | **Valor** |
| Objetivo | Gestionar cualquier módulo adicional por fuera de KoBoToolBox. Esto permite que los features solicitados puedan ser atendidos sin necesidad de modificar el código de KPI con los riesgos que implican |
| Incentivos | * Toda la lógica de incentivos se encontraría en esta capa de no tener un sistema de incentivos adicional. * Se recomienda simplificar con una API custom que envíe mensajes de incentivos a través de WebSockets. * Se puede enlazar fácilmente con la estructura modificada de ODKCollect/KoBoCollect. |
| Plataforma | React (all client side) |
| Deployment | API y FrontEnd se construyen juntos y generan un sólo container |

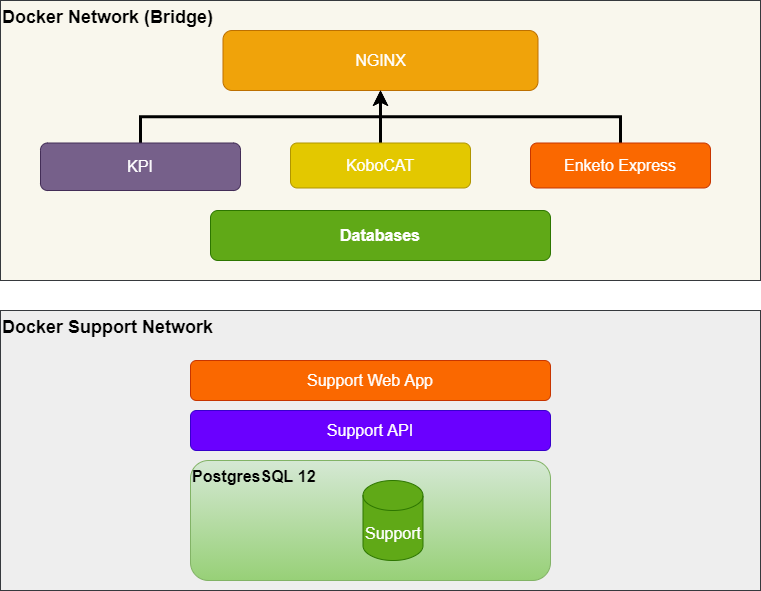
##### 

#### Capas de la Aplicación



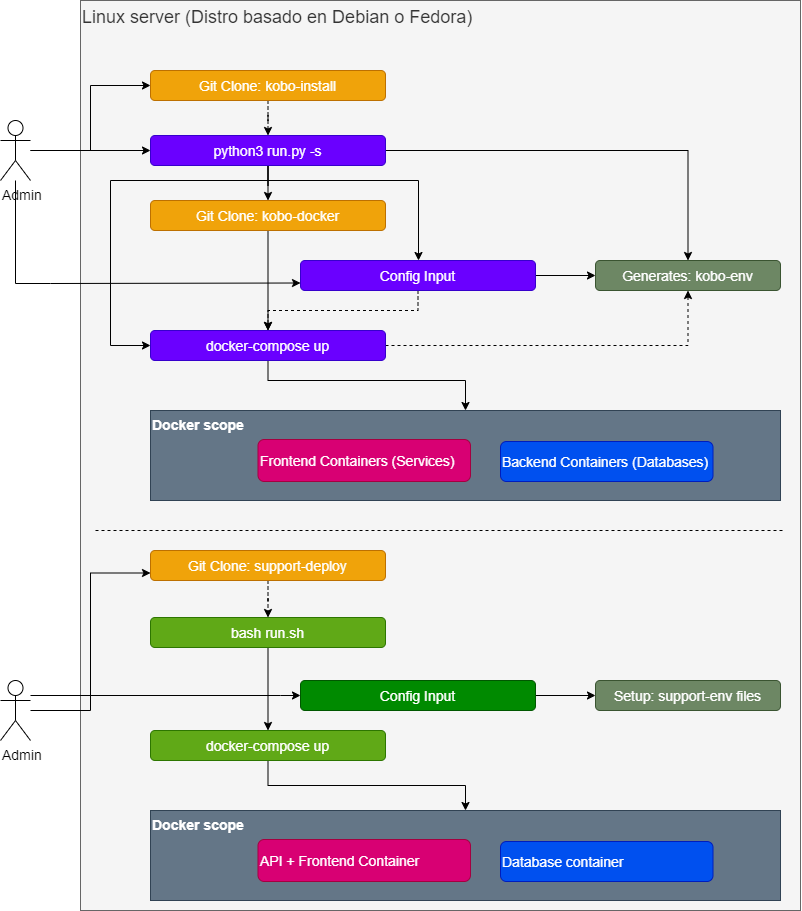
|  |  |
| --- | --- |
| **Concepto** | **Valor** |
| Capa: Aplicación | * Los proyectos KoBoToolbox no se modifican * Se crea un nuevo repo: Support-App   + API y FrontEnd conviven en el mismo repo   + Todos los cambios se manejan por Pull Request (en caso de usar Github) |
| Capa: KoBo-docker | * El proyecto KoBo-docker no se modifica * Se crea un nuevo repositorio: support-deploy que contiene los elementos de Docker y scripts de instalación |
| Capa: KoBo-install | * El proyecto KoBo-install no se modifica. * Se utiliza el proyecto support-deploy para crear scripts de instalación sobre docker. |

#### Diagrama de contenedores



|  |  |
| --- | --- |
| **Concepto** | **Valor** |
| Contenedores de servicios | * KPI * KoBoCAT * Enketo Express * **Support API**. Éste contenedor sería generado por el proyecto. |
| Base de datos | * Tipo: PostgresSQL. * Esquema: Support. * Locación:   + Sugerido: Tener un container propio basado en PostgresSQL 12.   + Segunda opción: Dentro del server usado por KoBoToolbox. |
| Docker Compose | * Se debe crear el servicio de Support API en su propio archivo docker-compose.yml en el proyecto support-deploy. * Support API no necesita estar bajo la misma red de KoBoToolbox. |
| Healthcheck | * Support API debe publicar un endpoint healthcheck básico que retorne un status 200 y un mensaje de texto “HEALTHY”. * Support API debe publicar un endpoint healthcheck de **base de datos** que retorne un status 200 y un mensaje de texto “DB HEALTHY”. |

#### Diagrama de Despliegue



|  |  |
| --- | --- |
| **Concepto** | **Valor** |
| Operador | Admin de servidor Linux |
| Roles | * Sudoer * Linux Group: Docker * Admin del socket de Docker * Acceso al repo de GIT |
| Acciones del Admin | * Git Clone de KoBo-docker * Git Clone de support-deploy. * Define los valores de configuración antes de ejecutar la instalación (run.sh). * Ejecución de bash run.sh. * Verifica que todos los servicios estén activos y funcionales a través de:   + docker ps   + docker logs   + python3 run.py -l   + python3 run.py -i   + docker exec -it <container> bash |
| Acciones de run.py | * Realiza git clone de KoBo-docker. * Crea el directorio KoBo-env en base a los valores de configuración provistos por el Admin. * Ejecuta las acciones de docker compose para crear (o recrear) el ambiente. |
| Acciones de run.sh | * Valida que los valores de configuración provistos existan. * Ejecuta docker-compose. * Realiza healthchecks de servicio API y conexión a base de datos. |

#### Flujo de Desarrollo

La definición de esta arquitectura es igual a la arquitectura A

[Ver a definición en Candidata A](#_nyqtcqv90liq)

#### Consideraciones Finales

##### Beneficios

* Toda la funcionalidad es manejada en una aplicación por fuera de KoBoToolbox lo cual no requiere ningún conocimiento de cómo programar en KT.
* No requiere una mobile app adicional (este acercamiento es el mismo que en Candidata A).
* Se puede escalar de manera libre e independiente.

##### Skills requeridos

* React
* Dotnet 5 (o Dotnet Core 3.1)
* Se requiere que el equipo de desarrollo cuente con conocimiento en Docker y Docker-compose para creación de scripts.

##### Riesgos

|  |  |
| --- | --- |
| **Tipo** | **Riesgo (Alto, Medio, Bajo)** |
| Support API no puede utilizar la base de datos de KoBoToolbox | Bajo |
| Cualquier error de programación (bug) durante operaciones requiere de los servicios de Nexion para la corrección/estabilización | Medio |
| El trabajo de modificación de docker-compose y puesta en marcha requiere comprensión completa del repo KoBo-docker | Alto |

##### Veredicto

* Esta arquitectura es la que mejor se adapta a las necesidades de independencia de plataforma, sin embargo el tiempo de desarrollo de la app Web podría ser más alto que customizar KoBoToolbox (Candidata A).
* Si se tiene un tiempo y presupuestos más holgados ésta arquitectura es viable.