Monitor de uso de la plataforma educativa CREA 2018-2021 y su puesta en producción

Natalia da Silva[1], Mauro Loprete[1], Oscar Montañés[1]

[1]Departamento de Métodos Cuantitativos, FCEA-UDELAR

Resumen

En este trabajo se presenta el monitor de uso de la plataforma de Contenidos y Recursos para la Educación y el Aprendizaje (CREA) utilizadas por el Plan Ceibal con datos de 2018 a 2021 y su puesta en producción. El entorno virtual de aprendizaje (CREA) permite gestionar cursos, crear o compartir materiales didácticos y trabajar colaborativamente en grupos. El monitor de uso de la plataforma educativa CREA es una herramienta que permite resumir los datos de uso de dicha plataforma por parte de los estudiantes de 4to 5to y 6to año de primaria entre 2018 y 2021 y permite un análisis a distintos niveles según el interés de las personas que usen el monitor. La información de uso en CREA se resume en base a indicadores de compromiso que sirven para comparar el uso de las plataformas incluso en años de pandemia donde el uso se incrementó considerablemente y algunos indicadores dejaron de ser útiles en ese contexto. El monitor es implementado en una aplicación web en base a Shiny disponible en un servidor de FCEA-UDELAR. La aplicación está diseñada para estar optimizado en términos de rendimiento y escalabilidad además que facilite futuras actualizaciones y mejoras, asegurando que la misma esté preparada para afrontar los desafíos cambiantes del entorno tecnológico y de comportamiento.

1 Introducción

En este trabajo se presenta el monitor de uso de la plataforma de Contenidos y Recursos para la Educación y el Aprendizaje (CREA) utilizadas por el Plan Ceibal con datos de 2018 a 2021 y su puesta en producción. El entorno virtual de aprendizaje (CREA) permite gestionar cursos, crear o compartir materiales didácticos y trabajar colaborativamente en grupos. El monitor de uso de la plataforma educativa CREA es una herramienta que permite resumir los datos de uso de dicha plataforma por parte de los estudiantes de 4to 5to y 6to año de primaria entre 2018 y 2021 y permite un análisis a distintos niveles según el interés de las personas que usen el monitor.

El monitor de uso presentado en este documento es parte de los resultados del proyecto titulado "Monitor y evaluación de uso de las plataformas educativas" financiado por ANII en el marco del Fondo Sectorial "Inclusión Digital: Educación con Nuevos Horizontes" - 2020 (proyecto) - Modalidad A (identificador del proyecto: FSED_2_2020_1_163528). El monitor de uso presentado en este documento tiene como principal antecedente el prototipo desarrollado en el marco del Fondo Sectorial de Investigación a partir de datos 2017 titulado "Nuevas tecnologías estadísticas para la evaluación y monitoreo de plataformas educativas". El prototipo resumía la información de uso de la plataforma CREA en base a indicadores de compromiso usando datos de 2015 y 2017 disponibles al momento de realizar dicho proyecto (Molina et al. 2021). El prototipo se implementó en una aplicación web que fue desarrollada principalmente en base a el paquete `shiny` (Chang et al. 2024) en el software estadístico R (R Core Team 2024). Dicha aplicación se encuentra disponible en http://164.73.240.157:3838/IESTA-CEIBAL/.

El presente proyecto pretende continuar con la misma línea de investigación extendiendo algunos aspectos claves para entender los determinantes del uso de las plataformas y la puesta en producción del prototipo desarrollado anteriormente para monitorear el uso de CREA.

Contar con herramientas que permitan monitorear el uso de las plataformas educativas de forma sistemática, estandarizada y sencilla ayuda a responder preguntas a distintos niveles de análisis, para diferentes actores del sistema educativo y son claves para el fortalecimiento de la educación tanto a distancia como presencial.

Dado que el uso de las plataformas educativas en el contexto de pandemia se incrementó considerablemente, es clave para el desarrollo de la aplicación web su optimización así como el uso de herramientas apropiadas para la gestión de los datos.

El nuevo monitor presentado en este trabajo está diseñado de una forma moderna y más robusta que el prototipo y la misma se adapta a nuevos desafíos incorporados por el mayor uso de las plataformas en los años de pandemia. Parte de las modificaciones técnicas en el diseño de la aplicación se justifican sobre la necesidad de manejar eficientemente un volumen de datos mucho mayor proveniente de los años de pandemia. El incremento exponencial de datos de uso de las plataformas requiere no solo una mejora en la capacidad de procesamiento, sino también una adaptación integral de la arquitectura para satisfacer las demandas de un volumen de datos sustancialmente mayor. Este proceso de migración no solo se enfoca en la optimización técnica, sino también en la capacidad de la aplicación para que en un futuro pueda ser adaptada y utilizada por un gran número de personas usuarias de la misma.

2 Datos

Para el desarrollo del monitor de uso de la plataforma CREA se cuenta con dos conjuntos de datos, el primero de ellos presenta el marco de alumnos y docentes para los grados 4to, 5to y 6to de primaria, donde las variables contienen información personal de cada usuario (fecha de nacimiento, sexo, departamento de residencia, etc) y relación con su ámbito de estudio (id del centro escolar, grado, clase, etc). Es importante resaltar que los datos utilizados están anonimizados tanto a nivel de individuos como de centros educativos.

El segundo conjunto de datos presenta registros de todas las actividades realizadas por cada usuario de la plataforma CREA en los días que hizo uso de ella entre 2018 y 2021. Este conjunto de datos tiene las mismas variables contenidas en el marco de alumnos y docentes agregando variables referidas a la actividad realizada en CREA (fecha del registro, comentarios posteados, trabajos enviados, etc), contando con un total de 56 variables.

En la Tabla 1 se muestran los registros de uso en CREA por estudiantes y docentes según año, se puede apreciar el salto en uso de la plataforma en los años de pandemia donde entre 2019 y 2020 casi se triplican los registros.

Tabla 1: Registros de uso en CREA por estudiantes y docentes según año

Año	Registros	Estudiantes	Docentes
2018	1.912801	98.054	3801
2019	2.652856	103.168	4474
2020	7.403004	109.019	5477
2021	8.607445	119.065	5299

3 Arquitectura

La nueva aplicación web al igual que el prototipo fueron diseñados usando Shiny (Wickham (2021)). Shiny es un marco que permite crear aplicaciones web usando código en R desarrollado por Posit (ex RStudio), su primera versión fue anunciada en 2012 por Joe Cheng. Actualmente también está disponible la versión de Shiny para Python presentada en 2022. Shiny permite crear desde aplicaciones web básicas sin necesidad de saber HTML, CSS o JavaScript pero también permite desarrollar aplicaciones web versátiles y personalizadas cambiando la interfaz de usuario (front end). A su vez Shiny se basa en la programación reactiva que permite la interactividad de la aplicación cuando los usuarios interactúan con la misma. En el componente de servidor la programación reactiva permite crear cualquier tipo de lógica en el back end.

En el diseño de la nueva aplicación web se tomaron en consideración algunas características deseables de la misma como ser: escalable, modularizable, suficientemente flexible para mantener, modificar y testear, así como ser reproducible y ser usada por muchos usuarios al mismo tiempo.

La migración de la aplicación Shiny en R hacia una versión más avanzada implica la adopción de una arquitectura robusta de tres capas, diseñada para potenciar la eficiencia, la escalabilidad y la modularidad del sistema (Figura 1). Esta estructura se organiza en capas de presentación, servicios y datos, cada una desempeñando un papel en la entrega de una experiencia de usuario optimizada y en la gestión eficiente de datos.

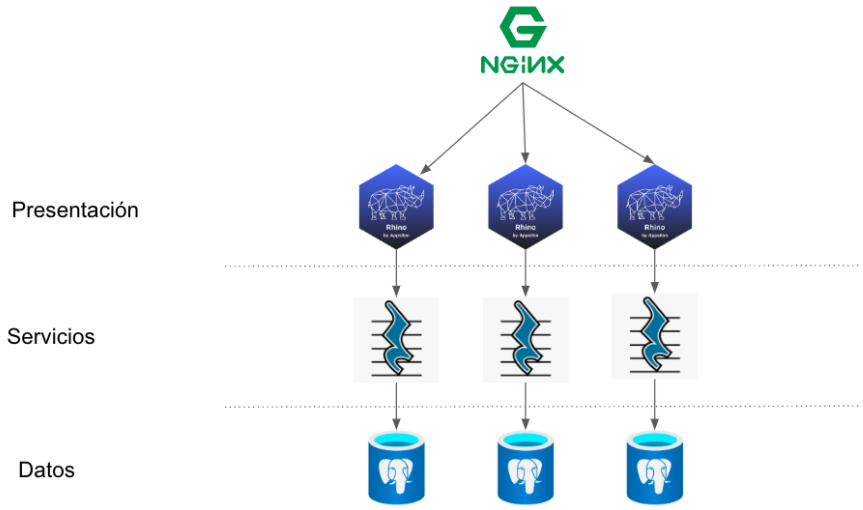

Figura 1: Arquitectura de tres capas

La adopción de esta arquitectura de tres capas no solo optimiza la aplicación en términos de rendimiento y escalabilidad, sino que también facilita futuras actualizaciones y mejoras, asegurando que la aplicación Shiny en R esté preparada para afrontar los desafíos cambiantes del entorno tecnológico y de comportamiento.

3.1 Capa de Presentación

En la capa superior, la interfaz de usuario y la presentación de datos adquieren protagonismo. Utilizando tecnologías provenientes de las aplicaciones Shiny, se busca mejorar la experiencia visual y la interactividad. Los elementos de la interfaz gráfica se diseñan para facilitar la comprensión de la información relacionada, proporcionando visualizaciones intuitivas y tableros dinámicos.

Una debilidad de la versión anterior (prototipo) del monitor educativo es que el diseño de la aplicación web no estaba modularizada. La modularización es un concepto extendido en ciencias de la computación desde el diseño de sistemas (Parnas 1972) hasta el diseño de software (Szyperski 1996). En términos generales se puede decir que la codificación por módulos es una estrategia de desarrollo de software que implica dividir un código en partes más pequeñas e independientes llamadas módulos. Cada módulo es una unidad independiente que realiza una tarea específica o implementa una funcionalidad particular y tiene una interfaz bien definida para interactuar con otros módulos. Esta técnica busca mejorar la estructura, el mantenimiento y la escalabilidad del código, al tiempo que facilita la colaboración entre las personas que lo desarrollan.

Se pueden enumerar algunas características y beneficios de trabajar con codificación por módulos:

1. **Descomposición de Tareas**: dividir un sistema en módulos permite atacar el problema general mediante tareas más pequeñas y por tanto más manejables. Cada módulo se centra en una función o característica específica, lo que simplifica el diseño y la implementación.
2. **Reutilización de Código**: los módulos son unidades independientes y autónomas por lo que pueden ser reutilizados en distintas partes de una aplicación o incluso en proyectos diferentes. Esto reduce la redundancia y facilita la creación de bibliotecas de funciones que pueden ser compartidas entre diferentes equipos o proyectos.
3. **Escalabilidad**: la modularidad facilita la escalabilidad del sistema. Se pueden agregar nuevos módulos o funcionalidades sin afectar las partes existentes, siempre y cuando se respeten las interfaces definidas.
4. **Mantenimiento**: la modularidad facilita el mantenimiento del código. Si se produce un error o se necesita una mejora en una función específica, solo se requiere modificar el módulo correspondiente sin afectar otras partes del sistema. Los módulos permiten crear pruebas unitarias específicas para cada uno de ellos, lo que facilita la identificación y corrección de errores. Esto facilita el mantenimiento y reduce el riesgo de introducir nuevos errores.
5. **Flexibilidad**: la arquitectura modular permite hacer cambios en un módulo sin afectar los demás. Esto es especialmente útil en proyectos grandes donde los requisitos pueden cambiar con el tiempo.
6. **Facilita la Colaboración**: la codificación por módulos permite que las personas que desarrollan puedan trabajar en paralelo en distintos módulos sin interferir en el trabajo de los demás. Cada módulo tiene interfaces bien definidas, lo que facilita la integración y la colaboración eficiente en equipos grandes.
7. **Testeo**: al tener unidades más pequeñas y enfocadas, el proceso de prueba y depuración se vuelve más manejable. Los módulos pueden ser probados de manera individual antes de la integración, lo que facilita la identificación y resolución de problemas.
8. **Legibilidad y Comprensión**: la codificación por módulos mejora la legibilidad del código al organizarlo de manera lógica y estructurada. Cada módulo se enfoca en una tarea específica, lo que hace que el código sea más comprensible y fácil de mantener.

La codificación por módulos fue introducida en Shiny 0.13 en 2016 y evolucionó en el tiempo modificando la estructura de la aplicación en su conjunto. Aunque se puede trabajar con módulos

usando Shiny directamente, en este proyecto se analizaron tres bibliotecas de R que permiten el desarrollo de aplicaciones Shiny de forma modular pero además que ofrecen otras ventajas relevantes para el proyecto. Las bibliotecas analizadas fueron: `rhino` (Żyła et al. (2024)), `golem` (Fay et al. (2023), Fay et al. (2021)) y `leprechaun` (Coene (2022)).

Comprender las principales características de estas tres propuestas y sus distinciones claves permitió identificar la más apropiada para este proyecto. Estas tres bibliotecas ofrecen beneficios únicos que se adaptan a necesidades específicas. A modo de resumen se puede mencionar que Rhino ofrece una opción robusta para aplicaciones empresariales de gran porte. Planteando un enfoque tal vez más estructurado para las pruebas que podría ser más acorde con la robustez y precisión organizativa. Por otro lado Golem se alinea bien con proyectos que valoran la creación rápida y casi sin esfuerzo de paneles de control, liberando tiempo para otras tareas importantes y las aplicaciones con `golem` están contenidas en un paquete. Finalmente Leprechaun se destaca para aquellos que prefieren un enfoque ligero y minimalista, integrándose sin problemas con proyectos Shiny existentes.

En base a las características y ventajas de las distintas propuestas se seleccionó a Rhino para el desarrollo de la aplicación ya que se adapta mejor a las necesidades de este proyecto y su futuro mantenimiento y escalabilidad. En lo que sigue se describen las principales características y ventajas de Rhino en base al material contenido en https://appsilon.github.io/rhino/.

3.1.1 Rhino

`Rhino` es un paquete de R que facilita la creación eficiente de aplicaciones Shiny de alta calidad a nivel empresarial. Algunas características destacadas por sus creadores son que promueve las mejores prácticas de ingeniería de software, modularización del código, pruebas exhaustivas y un diseño elegante de la interfaz de usuario. Ofrece un enfoque estructurado con énfasis en las mejores prácticas y herramientas de desarrollo. A su vez busca ahorrar tiempo de desarrollo y unificar la arquitectura de la aplicación. Es sencillo convertir una aplicación Shiny a proyectos de Rhino.

Algunas de las ventajas

- Rhino proporciona una estructura de archivos anidada para manejar aplicaciones más grandes y permite escalabilidad, modularización y una organización clara del código con una separación ordenada de responsabilidades.
- Acelera el desarrollo de aplicaciones Shiny mientras mejora la confiabilidad y mantenibilidad.
- Ofrece un conjunto de herramientas diseñadas para optimizar aplicaciones Shiny, simplificar el desarrollo y fomentar aplicaciones mantenibles y de alta calidad, especialmente en entornos empresariales.
- Rhino integra herramientas esenciales para el desarrollo de aplicaciones Shiny, mejorando la eficiencia y la calidad. Permite el uso de Cypress para pruebas de extremo a extremo y la integración de shinytest2 para garantizar aún más la confiabilidad de la aplicación.
- Incorpora una variedad de linters, incluidos los de R, JavaScript y Sass (CSS mejorado para estilizar tus aplicaciones), para asegurar código de alta calidad en diferentes lenguajes de programación.
- Apoya a las personas que desarrollan en la creación de código y estilos JavaScript, ofreciendo una configuración que empacará automáticamente scripts JavaScript y archivos Sass, simplificando su integración en proyectos.
- Cuenta con capacidades avanzadas de registro para una depuración efectiva, mejorando el proceso de desarrollo.

- Rhino ofrece soluciones basadas en la experiencia de la industria.
- La configuración de Rhino con GitHub Actions automatiza la ejecución de todas las pruebas y linters, garantizando una calidad de código consistente a través de la integración continua.
- Se destaca en el soporte de una amplia gama de pruebas junto con una configuración simplificada para pruebas de extremo a extremo, asegurando aplicaciones robustas y libres de errores.

Estructura de un Proyecto Rhino

Shiny tiene un potente modelo de programación reactiva y un conjunto completo de funciones para crear componentes en la interfaz de usuario o estructuras HTML personalizadas. Estas características hacen posible construir rápidamente aplicaciones interactivas impactantes, pero también pueden dificultar la prueba y reutilización del código.

Para abordar este problema es recomendable separar el código que depende de Shiny de la lógica que se puede expresar sin él. Esta división es crucial para construir aplicaciones robustas y mantenibles. Para respaldar esta separación, Rhino fomenta una estructura específica para las fuentes R de su aplicación:

- main.R: Punto de entrada a su aplicación.
- logic: Código de la aplicación independiente de Shiny.
- view: Módulos Shiny y código relacionado.

```
.
├── app
│   ├── js
│   │   └── index.js
│   ├── logic
│   │   └── __init__.R
│   ├── static
│   │   └── favicon.ico
│   ├── styles
│   │   └── main.scss
│   ├── view
│   │   └── __init__.R
│   └── main.R
├── tests
│   ├── cypress
│   │   └── integration
│   │       └── app.spec.js
│   ├── testthat
│   │   └── test-main.R
│   └── cypress.json
├── app.R
├── RhinoApplication.Rproj
├── dependencies.R
├── renv.lock
└── rhino.yml
```

Figura 2: Estructura inicial de una aplicación con Rhino

Esta estructura se justifica en el uso de los módulos de Shiny los que se encapsulan mediante el paquete `box` (Rudolph 2024) de forma consistente en toda la aplicación. Por ejemplo un archivo que es cargado con `box::use()` solo puede cargar otros módulos/paquetes con `box::use()` lo que significa que no se pueden usar las funciones `library()` o `source()` en la aplicación. Esta es una distinción importante de la estructura Shiny tradicional, donde simplemente estamos cargando `app.R` cuando se carga la aplicación.

En la Figura 3 se presenta una posible estructura de un módulo típico con Rhino.

```
box::use(
  shiny[moduleServer, NS, renderText, tagList, textInput, textOutput],
)
box::use(
  app/logic/messages[hello_message],
)

#' @export
ui <- function(id) {
  ns <- NS(id)
  tagList(
    textInput(ns("name"), "Name"),
    textOutput(ns("message"))
  )
}

#' @export
server <- function(id) {
  moduleServer(id, function(input, output, session) {
    output$message <- renderText(hello_message(input$name))
  })
}
```

Figura 3: Módulo típico con Rhino

Comunicación de módulos

A medida que la aplicación crece en tamaño y se vuelve más compleja, se pueden encontrar un número creciente de módulos Shiny distribuidos en varios niveles de profundidad. En este contexto resulta necesario compartir información entre los diversos módulos de la aplicación.

Hay varias maneras de atacar este problema, se presentan dos ejemplos. En el primer ejemplo (Figura 4), se pasa información (variables reactivas) de un módulo principal a un módulo secundario, y en el segundo ejemplo (Figura 5), se pasa información entre dos módulos hermanos.

En este ejemplo, tendremos dos módulos. El módulo principal cargará y procesará datos utilizando filtros y los pasará al módulo secundario, que mostrará una tabla para estos datos procesados.

En el segundo ejemplo, se puede considerar que se cuenta con un módulo de procesamiento de datos y un módulo de trazado, ambos funcionando como hermanos, con sus roles respectivos siendo procesar y mostrar datos en un gráfico dado. Estos módulos hermanos están anidados dentro de otro módulo, al que se denomina main.R, siendo el módulo principal dentro de una aplicación Rhino.

En el ejemplo mencionado anteriormente, se ha establecido una variable reactiva dentro del módulo de procesamiento de datos, responsable de contener los datos procesados generados por ese módulo. Posteriormente, este módulo devuelve dicha variable reactiva, que luego se pasa a la función del servidor del módulo de trazado. El módulo de trazado utiliza esta variable reactiva específica, que contiene los datos procesados, para crear visualizaciones.

3.2 Capa de de Servicios

En el medio de la arquitectura se encuentra la capa de servicios, encargada de la lógica de negocio y la gestión de operaciones. Aquí, se implementan las funciones que procesan y analizan

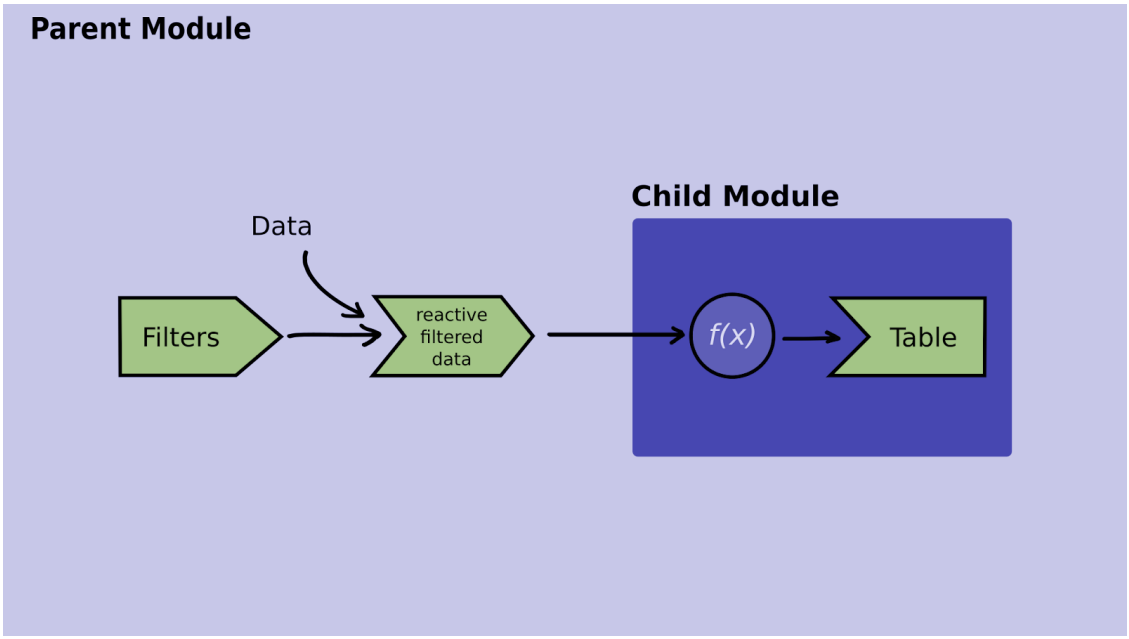

Figura 4: Comunicación entre un módulo principal y sus módulos secundarios

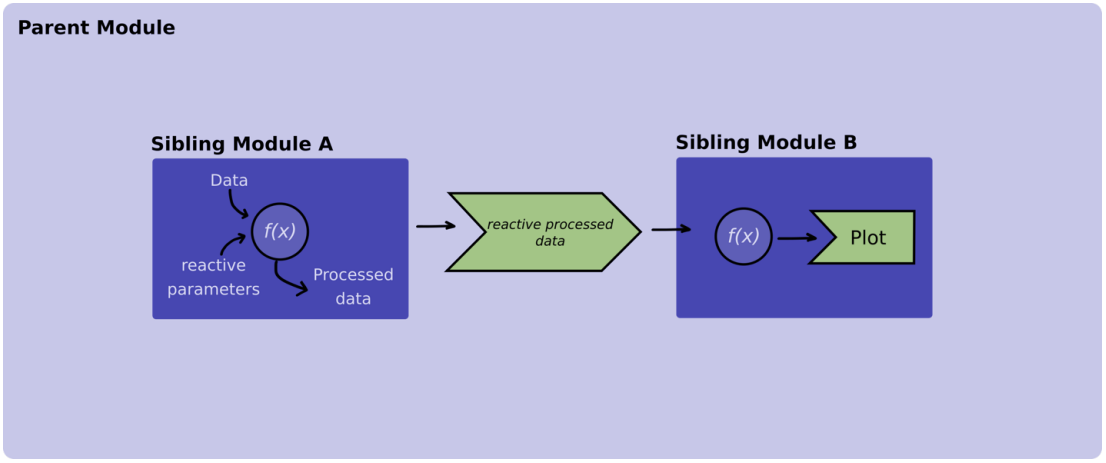

Figura 5: Comunicación entre dos módulos hermanos

los datos, incorporando las nuevas fuentes relacionadas con la pandemia. La lógica de la aplicación se fragmenta en servicios independientes y modulares, facilitando la mantenibilidad y la escalabilidad. Además, se integran servicios externos que pueden proporcionar datos adicionales o funcionalidades específicas, mejorando la capacidad de respuesta y adaptabilidad del sistema.

Para este caso dentro del proyecto se utilizó PostgREST, que es un servidor web independiente que convierte directamente tu base de datos PostgreSQL en una API RESTful. Las restricciones estructurales y los permisos en la base de datos determinan los puntos finales (endpoints) y operaciones de la API.

En esencia, PostgREST simplifica la creación de una interfaz de programación de aplicaciones (API) a partir de una base de datos PostgreSQL al proporcionar automáticamente puntos finales y operaciones basados en la estructura y permisos existentes en la base de datos. Esto facilita el acceso y la manipulación de los datos almacenados en PostgreSQL a través de solicitudes HTTP estándar, siguiendo los principios de diseño RESTful.

PostgREST se encargará de manejar estas operaciones, generando automáticamente los puntos finales (endpoints) y utilizando las restricciones y permisos definidos en la base de datos PostgreSQL para garantizar la seguridad y la coherencia de los datos. Este enfoque simplifica significativamente la creación y el mantenimiento de una API basada en PostgreSQL.

3.3 Capa de Datos

La base de la arquitectura reside en la capa de datos, donde se almacenan, gestionan y recuperan los datos necesarios para la aplicación. Dada la expansión del volumen de datos asociados con la pandemia, se implementan soluciones de almacenamiento eficientes y escalables.

Como base de datos se utilizó Postgres con tablas precalculadas. Las tablas precalculadas se refieren a tablas que contienen resultados previamente calculados o agregados en lugar de datos brutos. Estas tablas se crean para mejorar el rendimiento de las consultas y reducir el tiempo de respuesta al evitar realizar cálculos costosos cada vez que se necesita información específica. A continuación, se presentan algunos ejemplos de tablas precalculadas y su utilidad.

1. Tabla de Resumen o Agregación:
   - Propósito: Almacena resultados agregados o resúmenes de datos brutos.
   - Ejemplo: Una tabla que contiene el índice de compromiso a nivel de escuela de forma mensual en vez de tener que calcular el índice cada vez que se consulta.
2. Tabla de Índices:
   - Propósito: Mejora la velocidad de búsqueda.
   - Ejemplo: Una tabla que almacena ciertos campos junto con sus ubicaciones en la tabla principal para acelerar las búsquedas, similar a un índice.
3. Tabla de Unión:
   - Propósito: Almacena los resultados de operaciones de uniones complejas.
   - Ejemplo: Si tienes consultas frecuentes que involucran uniones complejas entre varias tablas, se puede crear una tabla precalculada que contenga los resultados de esas operaciones de unión.
4. Tabla de Datos Derivados:
   - Propósito: Almacena datos derivados que se utilizan con frecuencia.
   - Ejemplo: Una tabla que contiene la distancia entre dos ubicaciones geográficas precalculadas en lugar de calcularla cada vez que se realiza una consulta de distancia.
5. Tabla de Historial Agregado:
   - Propósito: Almacena información agregada a lo largo del tiempo.
   - Ejemplo: Una tabla que contiene sumas acumuladas mensuales de ciertos valores en lugar de calcular estas sumas cada vez que se consulta.

Algunas de las ventajas de las tablas precalculadas implican la mejora en el rendimiento, al tener resultados precalculados, las consultas pueden ejecutarse más rápidamente, ya que no es necesario realizar cálculos complejos en tiempo real. A su vez hay una mejor carga en el servicio ya que las operaciones costosas se realizan durante la carga inicial de los datos, reduciendo la carga en el servidor cuando se solicitan resultados. Las tablas precalculadas facilitan el mantenimiento ya que pueden simplificar el diseño y la administración de la base de datos al reducir la complejidad de las consultas en tiempo real. Finalmente la experiencia de las personas que usen la aplicación puede ser mejor ya que las consultas son más rápidas y los tiempos de respuesta inferiores lo que mejora la experiencia general. Cuando se trabaja con tablas precalculadas es necesario hacer actualizaciones periódicas para reflejar cambios en los datos subyacentes (en caso que los datos se actualicen regularmente). Esto puede hacerse mediante procesos de actualización programados. Hay que tener en cuenta que almacenar datos precalculados puede requerir más espacio en disco, especialmente si las tablas precalculadas contienen grandes cantidades de información. Finalmente es necesario considerar cuidadosamente qué consultas son más frecuentes y costosas en términos de rendimiento para determinar qué datos precalcular y almacenar.

En resumen, las tablas precalculadas son una estrategia efectiva para mejorar el rendimiento y la eficiencia de consultas en bases de datos, especialmente cuando ciertas operaciones o agregaciones son frecuentes y costosas.

A modo de ejemplo, en el monitor se utilizaron estas 2 consultas para precalcular donde va a ser computado un alumno en un momento dado (año o mes):

Por Año:

```
create table _precalculos_anio as (SELECT
        EXTRACT( 'year' FROM fecha) as anio,
        id_persona,
        departamento,
        id_centro,
        grado,
        COUNT(*) as cantidad,
        row_number() OVER (PARTITION BY EXTRACT( 'year' FROM fecha),
        id_persona ORDER BY COUNT(*) DESC, id_centro) as row_num
    FROM  datos_iesta_v2
    GROUP BY anio,id_persona,departamento,id_centro, grado
                )
```

Por Mes:

```
create table _precalculos_mes as (SELECT
        EXTRACT( 'year' FROM fecha) as anio,
        EXTRACT( 'month' FROM fecha) as mes,
        id_persona,
        departamento,
        id_centro,
        grado,
        COUNT(*) as cantidad,
        row_number() OVER (PARTITION BY EXTRACT( 'year' FROM fecha),
        EXTRACT( 'month' FROM fecha), id_persona ORDER BY COUNT(*) DESC,
        id_centro) as row_num
    FROM  datos_iesta_v2
    GROUP BY anio,mes,id_persona,departamento,id_centro, grado
```

En la Tabla 2 se presentan algunos ejemplos de tablas precalculadas que soportan a la aplicación.

Tabla 2: Tablas precalculadas

Tabla	Descripción
anio_departamento_centro	Muestra todos las combinaciones de departamento año y centro
datos_iesta_old	Antigua tabla con los datos raw
datos_iesta_v2	Tabla particionada e indexada con los datos raw de ceibal
rhino_density	Tabla que se utiliza para la visualización de la densidad
rhino_eng_dpto_anio_v6	Tabla con el índice de compromiso calculado por año por departamento
rhino_eng_indicador_false_anio_centro_grado_v6	Tabla con el índice de compromiso común a nivel de escuela-grado y de forma anual
rhino_eng_indicador_false_mes_centro_grado_v6	Tabla con el índice de compromiso común a nivel de escuela-grado y de forma mensual
rhino_eng_indicador_false_mes_centro_v6	Tabla con el índice de compromiso común a nivel de escuela y de forma mensual
rhino_eng_indicador_false_mes_departamento_v6	Tabla con el índice de compromiso común a nivel de departamento y de forma mensual
rhino_eng_indicador_true_anio_centro_grado_v6	Tabla con el índice de compromiso común a nivel de escuela-grado y de forma anual
rhino_eng_indicador_true_mes_centro_grado_v6	Tabla con el índice de compromiso cei a nivel de escuela-grado y de forma mensual
rhino_eng_indicador_true_mes_centro_v6	Tabla con el índice de compromiso cei a nivel de escuela y de forma mensual
rhino_eng_indicador_true_mes_departamento_v6	tabla con el índice de compromiso cei a nivel de departamento y de forma mensual
rhino_eng_nino_anio_v6	Índice de compromiso por niño por año
rhino_eng_nino_mes_v6	índice de compromiso por niño por mes

3.4 Despliegue y Mantenimiento

La aplicación y sus componentes están alojados en un servidor de la Facultad de Ciencias Económicas y de Administración de la Universidad de la República. Una vez finalizado el desarrollo de la aplicación, se procedió a su despliegue utilizando contenedores Docker para asegurar la portabilidad y escalabilidad del sistema (Kane y Matthias (2023)).

Docker (Merkel (2014)) es una plataforma de código abierto que permite a las personas que desarrollan empaquetar, distribuir y ejecutar aplicaciones en contenedores. Los contenedores son

unidades de software ligeros y portátiles que incluyen todo lo necesario para ejecutar una aplicación, como bibliotecas, herramientas y código. Son similares a las máquinas virtuales pero más rápidos y eficientes.

El uso de contenedores Docker simplifica el despliegue de aplicaciones en entornos de producción, asegurando que la aplicación se ejecute de la misma manera independientemente de las diferencias en la infraestructura subyacente.

Aunque configurar los contenedores puede ser desafiante debido a la gestión de dependencias a nivel de sistema operativo, paquetes de R y la configuración entre diferentes servicios y la aplicación, una vez establecida la configuración correcta, Docker facilita enormemente el despliegue y la gestión de aplicaciones en entornos de producción.

En el contexto de este proyecto, se encontró una solución que permite gestionar múltiples aplicaciones Shiny a nivel institucional, facilitando a los usuarios el mantenimiento y la actualización de las aplicaciones de manera sencilla.

La configuración estándar incluye dos contenedores principales: uno con las dependencias necesarias para la aplicación Shiny y otro con un servidor de RStudio para el desarrollo y mantenimiento de la aplicación. Esto permite a las personas usuarias realizar cambios y actualizaciones de manera controlada mediante credenciales y permisos específicos.

La comunicación y configuración de imágenes y contenedores se realiza a través de Docker Compose, una herramienta que facilita la manipulación y comunicación de diferentes contenedores mediante un archivo YAML legible y comprensible.

En la Figura 6 se muestra un ejemplo sencillo con los dos contenedores mínimos. En este ejemplo se especifica el tipo de contenedor y la versión a utilizar, con imágenes proporcionadas por el grupo Rocker, especializado en optimizar y construir imágenes de Docker para el entorno de R. Aunque para este proyecto construimos imágenes propias desde cero debido a la extensión y complejidad específica del proyecto.

```yaml
version: '3.6'
services:
  rstudio:
    container_name: rstudio-server
    image: rocker/rstudio:latest
    expose:
      - '8787'
    networks:
      - nginx_services
    volumes:
      - ./rstudio-server:/home/rstudio
  shiny:
    container_name: shiny-server
    image: rocker/shiny:latest
    expose:
      - '3838'
    networks:
      - nginx_services
    volumes:
      - ./shiny-app:/srv/shiny-server/shiny-app
networks:
  nginx_services:
    driver: bridge
```

Figura 6: Breve ejemplo de Docker Compose

4 Monitor de uso de CREA

En esta sección se describirá el contenido general del monitor de uso de la plataforma CREA para estudiantes con datos de 2018 a 2021. La aplicación actualmente se encuentra en el servidor de prueba en la Facultad de Ciencias Económicas y de Administración (https://test.fcea.edu.uy/shiny-ceibal/) y se está trabajando en un nuevo servidor con más prestaciones donde será migrada esta aplicación a futuro (https://shiny.fcea.udelar.edu.uy/shiny-ceibal).

4.1 Indicador de Compromiso

El monitor de uso de CREA intenta medir el uso de la plataforma a distintos niveles de análisis, para lo cuál se proponen distintos indicadores de compromiso y se seleccionan los más apropiados. Una diferencia importante con el prototipo de monitor que utilizaba datos de 2015 y 2017 es que en el monitor actual al incorporar datos provenientes de los años de pandemia hace que el indicador de compromiso utilizado anteriormente, propuesto en Marconi, Goyeneche, y Cobo (2017), no tenga sentido y sea necesaria un nuevo análisis y definición del mismo.

El nuevo índice de compromiso resume información de seis variables relevantes para los estudiantes de educación primaria y permite resumir la actividad a distintos niveles de análisis (clase, grado, escuela, departamento) y en distintos horizontes temporales (año, mes, semana). Dicho indicador está acotado inferiormente en 0, pero no posee una cota superior. En la Ecuación 1 se presenta el índice de compromiso.

$$IC_{it} = \frac{\sum_{j=1}^{6} log(x_{jit} + 1)}{log(2)} \tag{1}$$

donde j representa cada variable involucrada en el índice, i el estudiante y t el año. En términos diarios, si un usuario puede realizar una sola actividad contabilizable dentro del índice (es decir, perteneciente a las variables establecidas), entonces el índice puede aumentar por día $\log(1 + 1)$. La interpretación del valor del índice es la cantidad de días que el usuario entró y realizó al menos 1 actividad. Hemos de tener en cuenta que para los índices mensuales, por ejemplo, el valor puede superar la cantidad de días habilitados ya que el alumno en la realidad puede realizar más de una tarea diaria. Las variables consideradas en el índice de compromiso para estudiantes son: total de envíos calificables, total de subidas, total comentarios posteados, total comentarios posteados en envíos asignados, total días ingreso y total de envíos. En la Figura 7 se presenta la distribución del índice de compromiso para estudiantes según año, se puede observar una distribución similar para 2018 y 2019 con un valor promedio entorno a 15 mientras que para los años 2020 y 2021 la distribución presenta un corrimiento a la derecha con un valor promedio entorno a 35 lo cuál implica un incremento importante en el uso en los años de pandemia.

4.2 Shiny App

El monitor de uso de CREA está organizado en seis paneles: Inicio, Datos Nacionales, Datos por Escuela, Datos por Escuela y Grado, Datos por Clase y Reportes. Para esta aplicación se realizó un esquema de módulos por componente de la interfaz de usuario (UI), donde cada componente que se ve en la interfaz de usuario (tab, texto o gráfico) es un módulo de la aplicación usando el paquete `rhino`.

En la aplicación, el panel de **Inicio** se incluye una breve descripción del contenido de la aplicación.

En la Figura 8 se puede ver el panel de **Datos Nacionales** y en el panel lateral izquierdo las posibles usuarias del monitor pueden navegar por los distintos niveles de análisis de la aplicación.

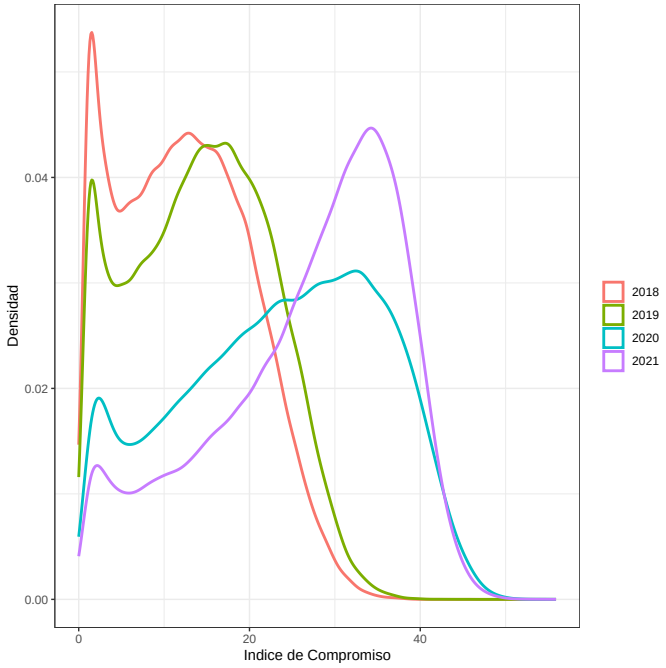

Figura 7: Indicador de Compromiso de Estudiantes en CREA según Año

A nivel Nacional hay tres filtros que las personas pueden seleccionar, Año (desde 2018 a 2021), Índice de Compromiso (dos opciones: Ceibal en Inglés y Común) y Medida (cuatro opciones: Media, Promedio 1er Cuartil y 3er Cuartil). En cuanto a la selección del Índice de Compromiso, la opción Ceibal en Inglés considera algunas variables diferentes en el cálculo del indicador (comentarios posteados, comentarios posteados en envíos asignados, total de comentarios posteados, total de envios, total de visitas, total de días de ingreso) consideradas más relevantes por el equipo de Ceibal en Inglés para medir el uso de la plataforma específico.

En este panel hay dos visualizaciones posibles, la densidad del índice de compromiso para los distintos grados a nivel nacional con la selección de alguno de los filtros antes mencionados, a su vez se muestra un mapa de Uruguay con los departamentos coloreados según el IC. Es importante resaltar que todas las visualizaciones estáticas contenidas en la aplicación fueron realizadas con el paquete `ggplot2` (Wickham (2016)) mientras que en caso de ser interactivas se utilizó el paquete `plotly` (Sievert (2020)).

En la Figura 9 se puede ver el panel de **Datos por Escuela**, en este panel se puede filtrar como en el panel anterior por año e índice de compromiso. A su vez se puede seleccionar departamento y escuela, siendo este último un número anonimizado. Una vez seleccionado los filtros se puede observar un gráfico de violín donde se muestra en el eje vertical la mediana del IC y en el eje horizontal los grados de primaria (4to, 5to y 6to). El violín representa la mediana del índice de compromiso para cada escuela perteneciente al departamento seleccionado y sus respectivos grados (4to, 5to, 6to). Con color rojo se muestra la mediana del IC de la escuela seleccionada para 4to, 5to y 6to. Esta visualización permite comparar cómo se encuentra la escuela seleccionada (según la mediana del IC) en los distintos grados según la distribución de la mediana del IC en dicho departamento. En este panel, se puede ver a su vez la distribución mensual del IC mediano para la selección realizada y cómo se compara con la evolución del IC mediano para las escuelas pertenecientes al departamento seleccionado. También se presenta una tabla con la cantidad total

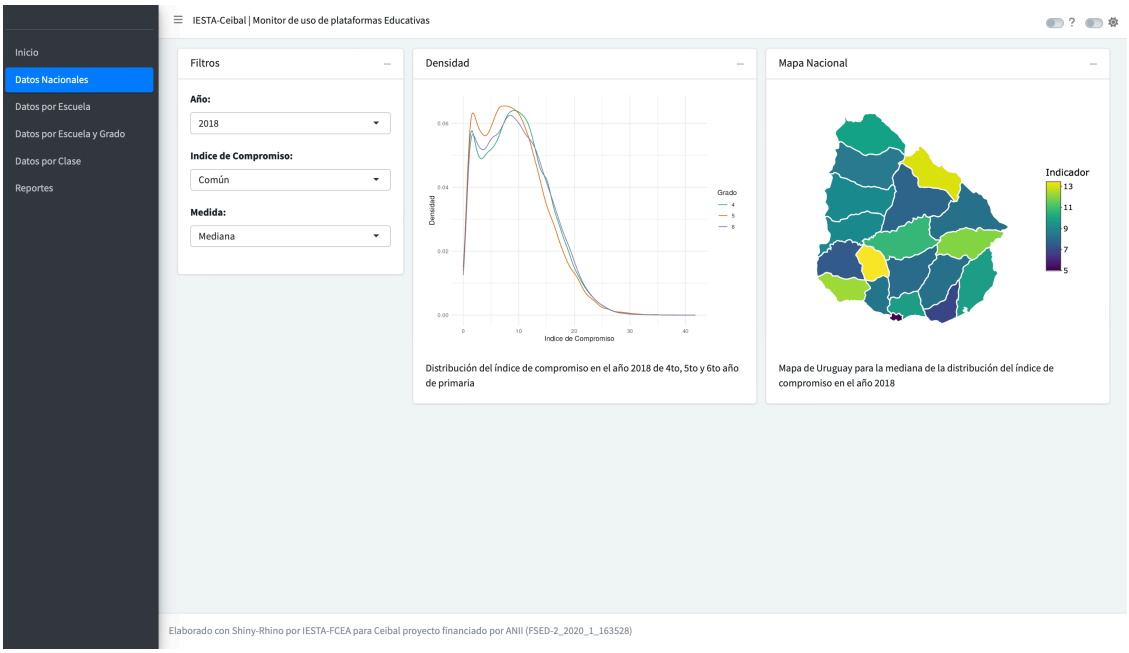

Figura 8: Monitor de uso de CREA 2018-2021, Datos Nacionales

de escuelas del departamento seleccionado en 4to, 5to o 6to, el total de alumnos, la mediana del IC y el ranking a nivel de departamento de esa escuela grado. Finalmente se presenta el gráfico de serie temporal que muestra la evolución a lo largo del año de las 6 variables que componen el índice de compromiso. En el mismo se muestra en el eje vertical el promedio normalizado de las dimensiones que componen el IC y en el eje horizontal los meses del año seleccionado. Con color azul se muestra la escuela seleccionada. Con color rojo se muestran los valores promedios del departamento seleccionado.

En la Figura 10 se puede ver el panel de **Datos por Escuela y Grado**, en este panel al igual que en el anterior se puede seleccionar el año, el índice de compromiso, el departamento y la escuela. En este panel hay 5 visualizaciones que resumen la información del IC mensual y por grado para la selección realizada y su comparación con el IC a nivel del departamento seleccionado. El primer gráfico se presenta en el eje vertical la mediana del IC y en el eje horizontal los meses del año con la evolución mensual de la mediana del IC de la escuela seleccionada, para 4to, 5to y 6to grado de primaria. El color azul muestra la evolución promedio, de las medianas de compromiso a nivel del departamento seleccionado. Por otro lado se presenta un gráfico de cajas donde en el eje vertical se presenta el valor del IC para cada alumno y en el eje horizontal los grados de primaria (4to, 5to y 6to). Se observa la distribución del IC en 4to, 5to y 6to grado de primaria, de la escuela seleccionada. Cada punto representa el valor del IC de un alumno particular. Adicionalmente se presenta la evolución mensual de los componentes del IC en promedio, el promedio de componentes del IC por grado y el compromiso mediano por grado. Por más detalles explorar la aplicación.

En la Figura 11 se presenta el panel de **Datos por Clase**, los filtros que se pueden seleccionar son, año índice de compromiso, departamento, escuela, grado y clase. En este nivel de análisis se puede observar algún resumen de compromiso a nivel de estudiantes. Se presenta un gráfico de caja donde en el eje vertical se muestra el valor del IC, mientras en el eje horizontal el grado de primaria elegido. Se muestra el valor del IC para cada alumno seleccionado según los filtros. También se presenta el gráfico de serie temporal que presenta en el eje vertical el valor del IC y en

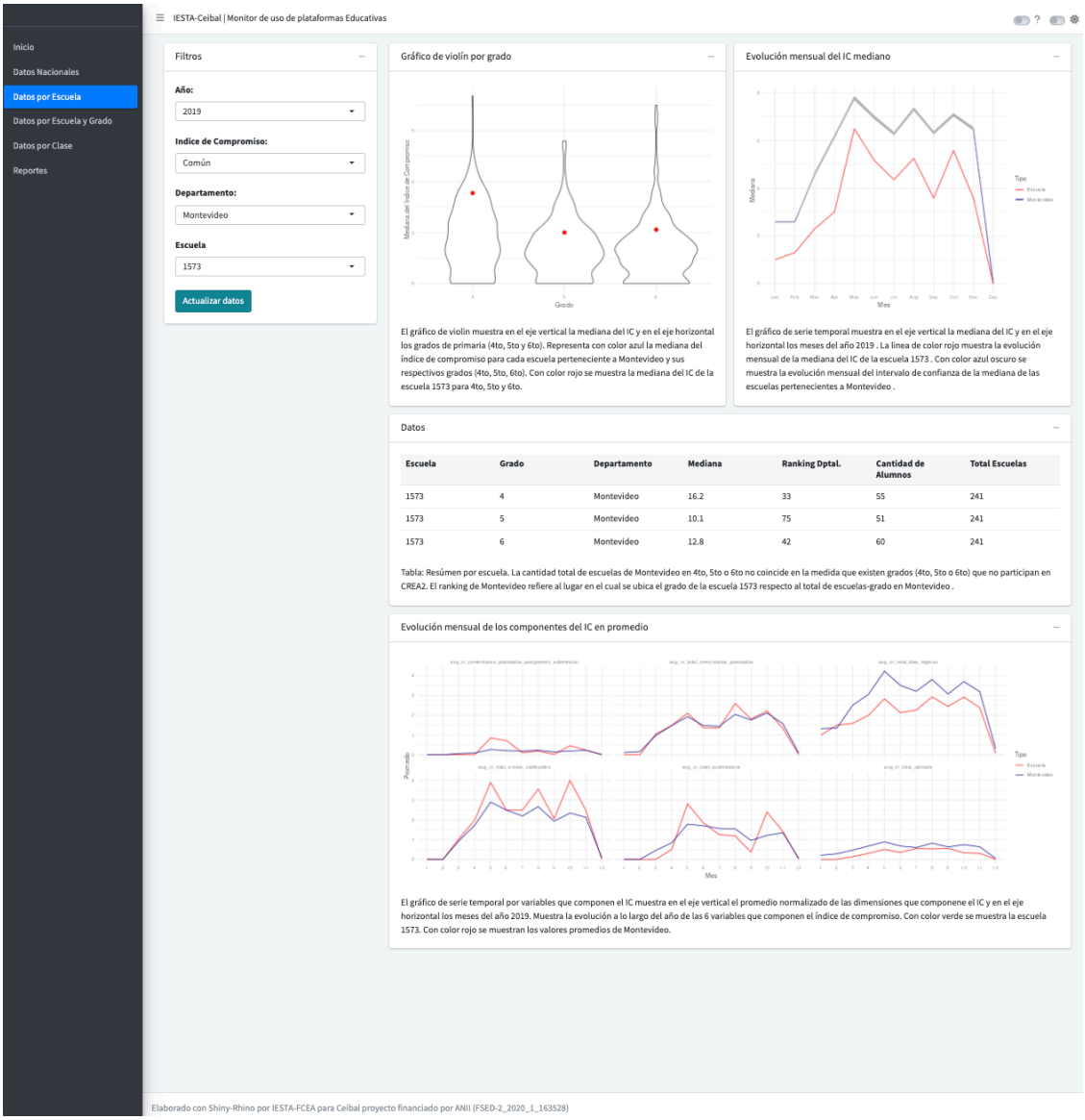

Figura 9: Monitor de uso de CREA 2018-2021, Datos por Escuela

el eje horizontal los meses del año. Cada línea roja punteada representa un alumno, mientras la línea azul representa el valor medio de la clase en cada mes. El área gris denota 2 errores estándar de la media. Se muestra a todos los alumnos de la clase seleccionada y su evolución a lo largo del año. Finalmente se presenta un gráfico de caja con las variables que componen el IC. en el eje vertical se presenta el promedio normalizado (estandarización) de las variables que componen el índice de compromiso y el eje horizontal representa a las 6 variables del índice de compromiso en el año seleccionado.

En el último panel de la aplicación se presenta un reporte que resume todas las selecciones realizadas en el monitor y dicho reporte se puede exportar a distintos formatos.

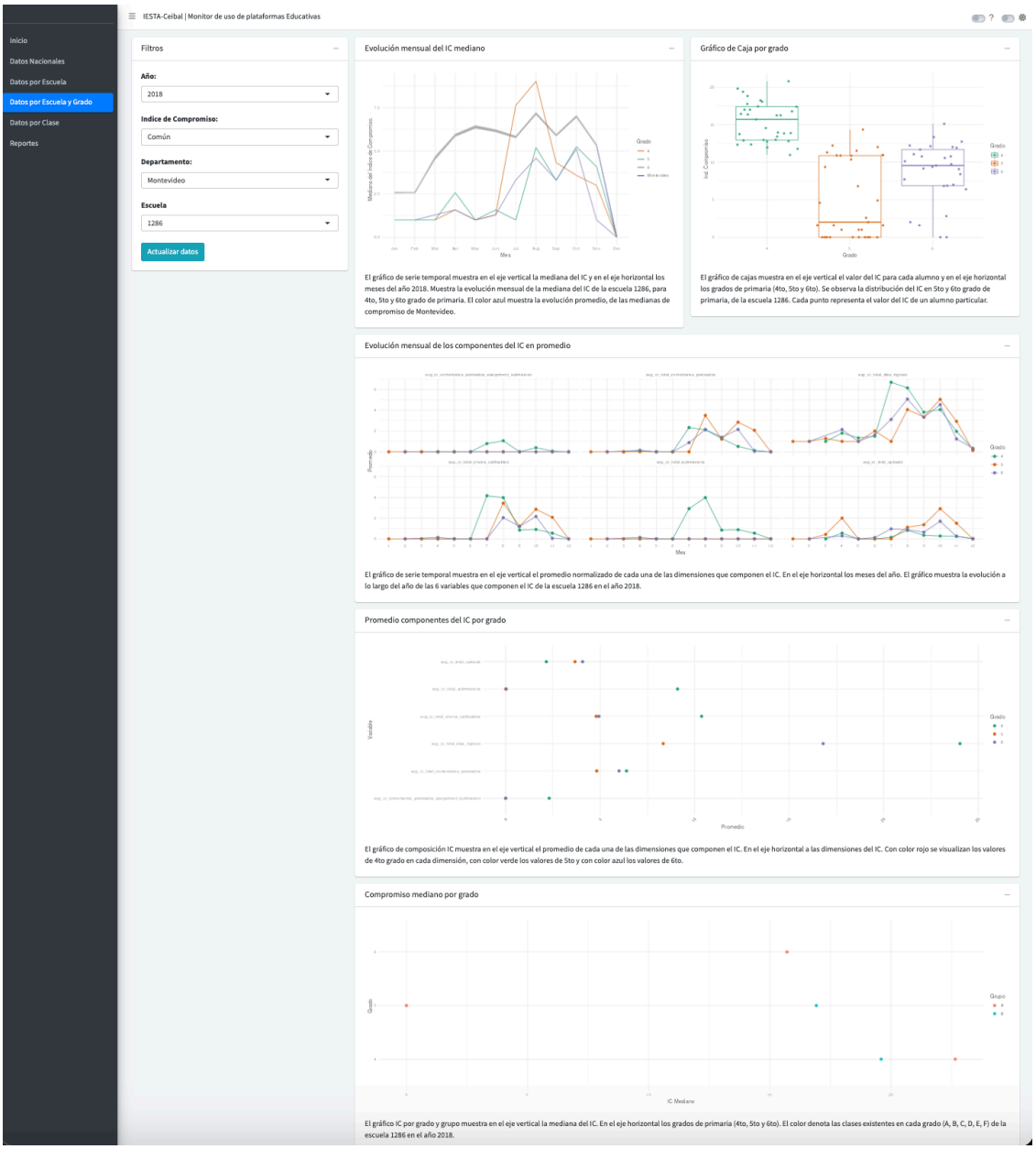

Figura 10: Monitor de uso de CREA 2018-2021, Datos por Escuela y Grado

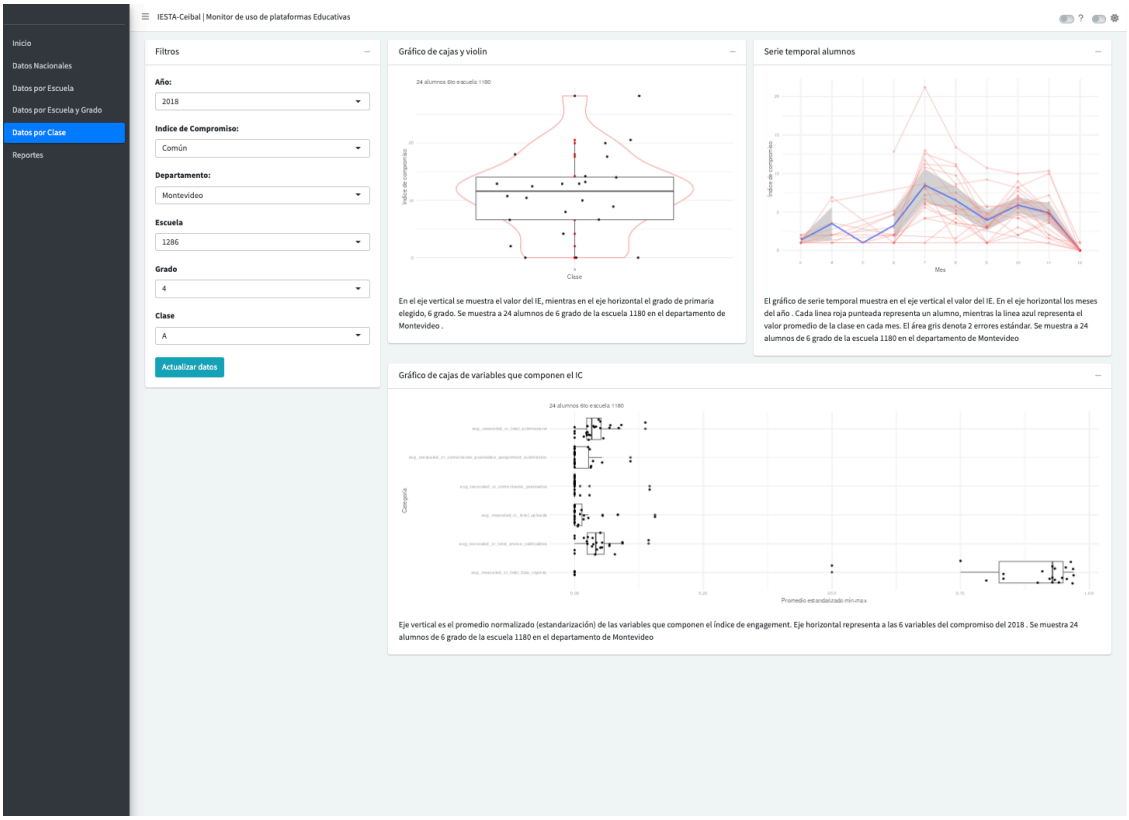

Figura 11: Monitor de uso de CREA 2018-2021, Datos por Clase

5 Comentarios Finales

En el presente documento se presentó el monitor de uso de la plataforma CREA y algunos detalles técnicos de su puesta en producción.

La migración de la aplicación Shiny en R hacia una versión más avanzada implicó la adopción de una arquitectura robusta de tres capas, diseñada para potenciar la eficiencia, la escalabilidad y la modularidad del sistema. La estructura se organiza en capas de presentación, servicios y datos, cada una desempeñando un papel en la entrega de una experiencia de usuario optimizada y en la gestión eficiente de datos.

El monitor está desarrollado de forma modular lo que mejora la estructura, el mantenimiento y la escalabilidad del código, al tiempo que facilita la colaboración entre las personas que lo desarrollan.

Una vez finalizada la etapa de desarrollo de la aplicación, se procedió a su despliegue utilizando contenedores Docker para asegurar la portabilidad y escalabilidad del sistema.

El producto final es una herramienta útil para analizar los datos de uso de la plataforma CREA y permitir la toma de decisiones informadas en base a ella a distintos niveles de análisis y para distintos roles dentro de la institución educativa. Este monitor puede ser extendido para otras plataformas definiendo indicadores específicos de uso en cada caso y niveles de análisis adicionales si el problema así lo requiere.

Como trabajo futuro sería deseable contar con datos a tiempo real o al menos que sea posible actualizaciones regulares que permitan realmente utilizar esta como una herramienta de consulta para la toma de decisiones basadas en evidencia.

Bibliografía

Chang, Winston, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, Jonathan McPherson, Alan Dipert, y Barbara Borges. 2024. *shiny: Web Application Framework for R*. https://CRAN.R-project.org/package=shiny.

Coene, John. 2022. *leprechaun: Create Simple 'Shiny' Applications as Packages*. https://CRAN.R-project.org/package=leprechaun.

Fay, Colin, Vincent Guyader, Sébastien Rochette, y Cervan Girard. 2023. *golem: A Framework for Robust Shiny Applications*. https://CRAN.R-project.org/package=golem.

Fay, Colin, Sébastien Rochette, Vincent Guyader, y Cervan Girard. 2021. *Engineering production-grade shiny apps*. Chapman; Hall/CRC.

Kane, Sean, y Karl Matthias. 2023. *Docker: Up and Running*. O'Reilly Media.

Marconi, Cecilia, Juan José Goyeneche, y Cristóbal Cobo. 2017. «When teachers and machines achieve the best combination: A national comparative study of face-to-face and blended teaching and learning».

Merkel, Dirk. 2014. «Docker: lightweight linux containers for consistent development and deployment». *Linux journal* 2014 (239): 2.

Molina, Federico, Natalia da Silva, Ignacio Alvarez-Castro, y Juan José Goyeneche. 2021. «Evaluación y monitoreo de plataformas educativas». *Serie Documentos de Trabajo; 2/21*.

Parnas, David Lorge. 1972. «On the criteria to be used in decomposing systems into modules». *Communications of the ACM* 15 (12): 1053-58.

R Core Team. 2024. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Rudolph, Konrad. 2024. *box: Write Reusable, Composable and Modular R Code*. https://CRAN.R-project.org/package=box.

Sievert, Carson. 2020. *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman; Hall/CRC. https://plotly-r.com.

Szyperski, Clemens. 1996. «Independently extensible systems-software engineering potential and challenges». *Australian Computer Science Communications* 18: 203-12.

Wickham, Hadley. 2016. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

———. 2021. *Mastering shiny*. " O'Reilly Media, Inc.".

Żyła, Kamil, Jakub Nowicki, Leszek Siemiński, Marek Rogala, Recle Vibal, Tymoteusz Makowski, y Rodrigo Basa. 2024. *rhino: A Framework for Enterprise Shiny Applications*. https://CRAN.R-project.org/package=rhino.