



C.P.R. Liceo "La Paz"

Desarrollo De Aplicaciones Multiplataforma

Proyecto Fin de Ciclo: StayRPE

Autor: Laura Lodeiro Casas
Tutor: Jesús Ángel Pérez Roca



Contenido

01

Introducción

¿De qué trata StayRPE?

02

Motivación

¿Por qué nació StayRPE?
¿Qué problema detecté?

03

Objetivo

¿Qué hueco real existe y
cómo lo resuelve StayRPE?

04

Tecnologías

¿Qué tecnologías hay
detrás?

05

Conclusión

¿Qué valor y aprendizaje
deja StayRPE?

01 ✨ Introducción ✨

¿De qué trata StayRPE?



Stay
RPE

02 ✨ Motivación ✨

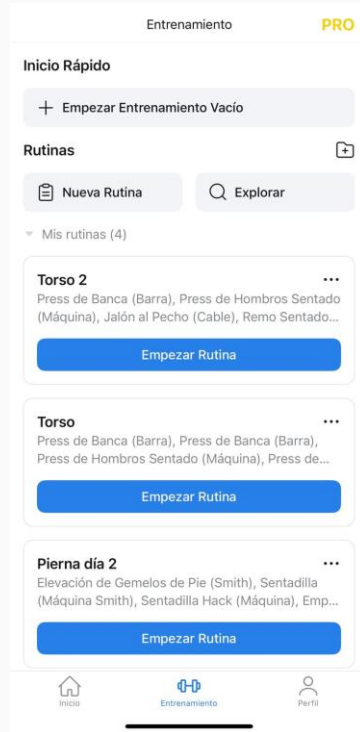
¿Por qué nació StayRPE? ¿Qué problema detecté?



El auge en ambos segmentos es imparable: el sector fitness, manteniendo su actual ritmo de actividad, superará los 1.500 millones de euros en 2025 en España, según las previsiones de **Intelligence 2P**. A nivel global, el crecimiento se estima por encima de los 300.000 millones de dólares en 2035. Estas cifras reflejan un incremento del peso del deporte en la economía nacional y que responde, en gran medida, a una mayor conciencia por compensar los –no tan buenos y tampoco tan pocos– efectos de las vidas urbanitas.

03 ✨ Objetivo ✨

¿Qué hueco real existe y cómo lo resuelve StayRPE?



Captura de pantalla Hevy –
Crear rutinas



Captura de pantalla Hevy –
Registrar entrenamientos



Objetivos StayRPE

Planificar rutinas cerradas durante semanas o meses.

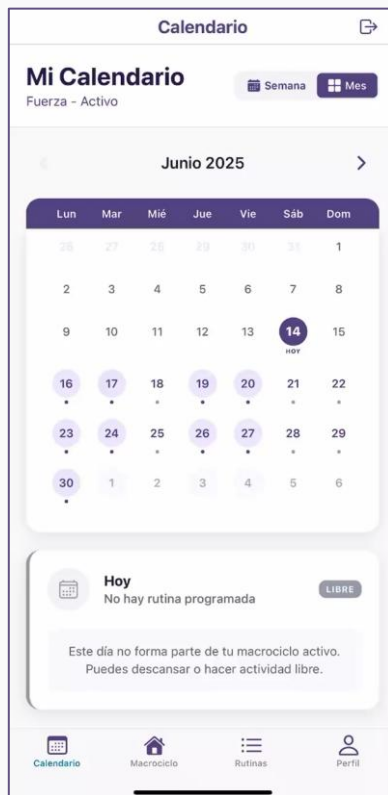
Modificar fácilmente cualquier día concreto.

Configurar la duración de cada microciclo sin complicaciones.

Registrar y guardar la intensidad real de cada serie y ejercicio.

Un macrociclo, es decir, una planificación a largo plazo, está formado por microciclos, que normalmente duran una semana.

Ejemplo: 1 mes (1 macrociclo) puede estar formado por 4 microciclos de una semana (4 semanas).

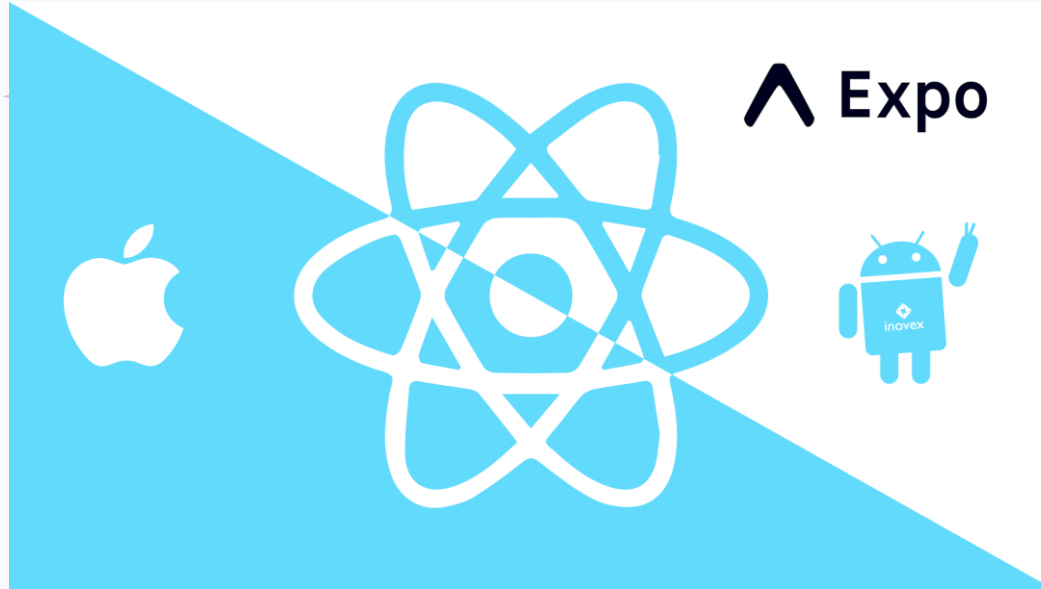


04 ✨ Tecnologías ✨

¿Cómo está hecha StayRPE y por qué usé esas tecnologías?




Frontend: React Native + Expo





¿Cómo hago que el calendario sea instantáneo, aunque tenga cientos de días?

Problema: Si el  macrociclo es muy largo, consultar cada día uno a uno hace que la app vaya lenta (esperas cada vez que tocas un día).

Solución: Cargar todos los días personalizados al inicio en una estructura Set().

Ventaja: La consulta de si un día está personalizado se hace al instante, aunque haya muchísimos días.

```
const [customizedDays, setCustomizedDays] = useState<Set<number>>(new Set());
```

¿Cómo consigo que la app sea rápida y no recargue todo innecesariamente?

Problema: En muchas apps, al volver al calendario se recarga toda la pantalla, aunque solo haya cambiado un detalle.

Solución: Implementé actualizaciones selectivas usando hooks y dependencias concretas, para que solo se refresquen los datos que han cambiado realmente.

Ventaja: El usuario vuelve al calendario y ve los cambios al instante, sin recargas completas ni esperas innecesarias. La experiencia es mucho más ágil y directa.

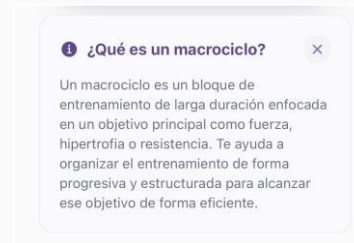
```
useFocusEffect(
  React.useCallback(() => {
    if (token && activeMacrocycle) {
      Promise.all([
        loadDayPlans(activeMacrocycle.id),
        loadCustomizedDays(activeMacrocycle.id)
      ]).catch(error => {
        console.error(' Error recargando datos especificos:', error);
      });
    } else if (token && !activeMacrocycle && !loading) {
      setDayPlans([]);
      setCustomizedDays(new Set());
    }
  }, [token, activeMacrocycle, loading])
);
```

¿Cómo consigo que la app recuerde la situación de cada usuario?

Problema: Muchas apps muestran lo mismo a todos, sin importar en qué punto esté cada uno.

Solución: Guardo el estado y preferencias de cada usuario por separado con AsyncStorage.

Ventaja: La app se adapta sola y muestra solo lo que corresponde a cada persona, haciendo la experiencia mucho más personalizada y cómoda.





Backend:Spring Boot + MySQL

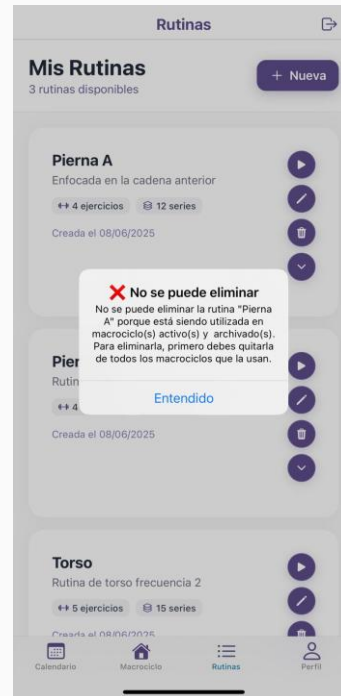


¿Cómo evito errores confusos y garantizo una experiencia clara para el usuario?

Problema: MySQL solo da errores genéricos como "FK constraint violation", que no ayudan al usuario a entender qué pasa.

Solución: Antes de hacer cambios críticos (como editar una rutina), hago validaciones manuales para comprobar si esa rutina está siendo usada.

Ventaja: Si intentas editar una rutina que está en uso, te muestro un mensaje claro.

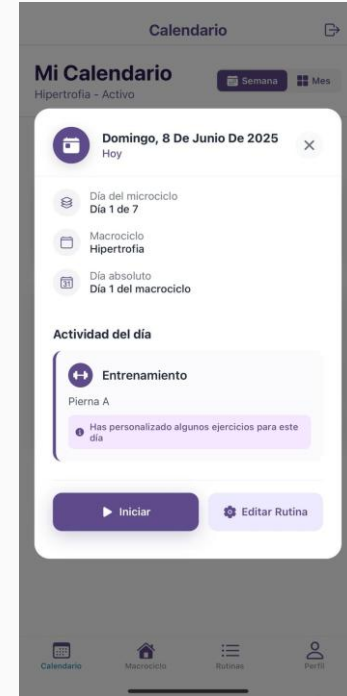


¿Cómo permito personalizar cualquier día sin romper la rutina original?

Problema: Si cambias el peso de una serie solo para el día 15, modificar la rutina original afecta al resto de días y usuarios.

Solución: Uso una tabla de customizaciones que solo guarda los cambios específicos de cada día.

Ventaja: El sistema combina los datos base con las customizaciones al vuelo, así puedes personalizar cualquier serie de cualquier día sin alterar la rutina original.





¿Cómo garantizo que los cambios se hagan completos o no se hagan?

Problema: Operaciones como eliminar un macrociclo implican varios pasos (borrar personalizaciones, desasociar entrenos, eliminar planes de día, etc.) y si algo falla a mitad, los datos pueden quedar a medias.

Solución: Uso transacciones atómicas con `@Transactional` en Spring Boot. Todos los cambios se mantienen "en borrador" hasta que todo termina bien.

Ventaja: Si algo falla a mitad de proceso, se deshace automáticamente todo lo anterior y la base de datos queda igual que al principio.

```
@DeleteMapping("/{id}") no usages LauraDev *
@Transactional
public ResponseEntity<> deleteMacrociclo(@PathVariable Long id) {
    logger.info("Eliminando macrociclo con ID: {}", id);

    Usuario usuario = getCurrentUser();
    if (usuario == null) {...}

    try {...} catch (Exception e) {
        logger.error("Error al eliminar macrociclo con ID: {}", id, e);
        return ResponseEntity.status(500).body(Map.of("error", "Error al eliminar el macrociclo"));
    }
}
```

05 ✨ Conclusión ✨

¿Qué valor y aprendizaje deja StayRPE?



Gracias por su
atención

