Escuela
Politécnica
Superior

IASport

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Pablo Corral Llorca

Tutor/es:
Domingo Gallardo López

Universitat d'Alacant
Universidad de Alicante

Junio 2018

Índice

Índice de ilustraciones

Justificación y objetivos generales

En el mercado existen numerosas aplicaciones relacionadas con el deporte. Todas ellas poseen rutinas formadas por un grupo de ejercicios, que se deben realizar un número de veces (repeticiones), en un día de entrenamiento (sesión) y según unas reglas concretas en función del tipo de entrenamiento.

Una vez terminada la sesión, la única información con la que contamos se obtiene mediante preguntas del tipo: cuánto le ha costado, ha terminado su ejercicio, ha realizado el número de repeticiones estipuladas de cada una de las series, etc.

Estas preguntas no son capaces de captar información esencial sobre aspectos más importantes como la correcta ejecución del ejercicio realizado siguiendo la técnica reglamentaria y sobre el cumplimiento o no de los tiempos de forma estricta.

De esta necesidad nace IASport, una aplicación que permite no sólo seguir una rutina de entrenamiento, sino que también monitoriza cada uno de los ejercicios de la misma garantizando que el movimiento se esté realizando rigurosamente y entre los límites de tiempo que marca las reglas de cada tipo de entrenamiento. De esta forma la progresión será mayor y evitará lesiones importantes causadas por una mala técnica.

Para ello, esta aplicación cuenta con un potente algoritmo capaz de controlar la realización de los diversos ejercicios mediante la captación y el análisis de datos en tiempo real. Tras dicho análisis, se extrae información valiosa para su posterior visualización en gráficas que reflejarán su progresión, tipos de errores cometidos, número de repeticiones realizadas correctamente/erróneamente, número de repeticiones por tipo de entrenamiento y un largo etcétera.

Asimismo, resaltar otro elemento diferenciador de la aplicación como es su flexibilidad, ya que puede crear o incorporar ejercicios personalizados o individualizados para cada necesidad.

Esta flexibilidad permite ampliar su aplicación inicial incorporando también el campo de la rehabilitación, donde la personalización o adecuación del ejercicio a cada paciente es crucial (distintos ejercicios para cada lesión). IASport permitiría a personas que se encuentren en este

proceso, seguir sus ejercicios de rehabilitación sin necesidad de desplazarse a un centro o tener que contar con un fisioterapeuta, lo que supone un ahorro tanto económico como de tiempo.

Finalmente, concluir que el éxito, tanto de un deportista como de un paciente que se encuentre en un proceso de rehabilitación, se basa en la exactitud con la que se realice los ejercicios, haciendo que se lleven a cabo no sólo de forma continuada sino también mediante una técnica rigurosa.

Agradecimientos

Para comenzar, me gustaría agradecer a mi tutor, Domingo Gallardo López, por haberme guiado a lo largo de todo el proceso, por haberme apoyado desde el minuto uno y, ante todo, por haberme inspirado y motivado tantísimo como profesor en todas y cada una de sus clases.

También a mi familia, en especial a mis abuelos, a mi pareja y a mis amigos, por su apoyo y haber creído en mi desde el primer momento.

Asimismo, especial mención a Yolanda Villacampa Esteve, del departamento de matemática aplicada de la EPS de la Universidad de Alicante que me dio una amplia visión de los diferentes algoritmos que existen en el mercado.

Por último, quería agradecer a la Universidad de Alicante, a la Escuela Politécnica Superior, así como también a todos los docentes que me han impartido.

Gracias.

Citas

“Projects we have completed demonstrate what we know - future projects decide what we will learn.”

— Dr Mohsin Tiwana

“An idea is just a dream until you code it”

— Unknown

“A lot of times, people don’t know what they want until you show it to them”

— Steve Jobs

Resumen

IASport nace para paliar la carencia que poseen la mayoría de aplicaciones destinadas a la realización de deporte como es la capacidad de poder detectar si el ejercicio se está realizando correctamente. IASport resuelve este problema con un potente algoritmo que monitoriza diversos aspectos de la realización de los ejercicios mediante la captación y el análisis de datos en tiempo real. Tras el análisis de estos datos, se genera un informe con el resultado del entrenamiento. Toda la información es almacenada para su posterior visualización en gráficas que reflejarán una visión global de la evolución del usuario, tipos de errores de ejecución, número de repeticiones realizadas correcta o incorrectamente, etc. Esta aplicación además de contar con gran cantidad de rutinas a las que poder apuntarse, también permite añadir ejercicios nuevos y personalizados adaptándose casi a cualquier necesidad. Finalmente destacar que, gracias a esa flexibilidad, esta aplicación puede también encajar en otros ámbitos más allá del deportivo como es el caso de la rehabilitación, ya que garantiza que el ejercicio se esté realizando correctamente según la técnica rigurosa que los expertos dictaminan.

1. Estado del arte

Previo al trabajo, se ha realizado una búsqueda exhaustiva tanto de estudios como aplicaciones en el mercado que tengan un enfoque similar a IASport.

1.1 Estudios

1.1.1 imWell[1]

imWell es un sistema móvil para la evaluación de la respuesta física a transiciones posturales. Parte de la base de que todo cambio postural inicia una respuesta física dinámica que puede ser usado como indicador del estado general de salud de una persona.

Hace uso de un componente externo (Zephyr BioHarness 3), monitor fisiológico que continuamente envía actividad a un Smartphone personal. Los datos capturados vienen dados directamente desde el acelerómetro y el ritmo cardíaco. El Smartphone procesa dichos datos en tiempo real para reconocer transiciones de posturas y caracterizar como ese cambio en las posturas afecta directamente al ritmo cardíaco.

El objetivo principal de este estudio es testar la hipótesis en la que un sujeto al que se le monitoriza tanto su ritmo cardíaco, así como también su actividad física durante el suficiente tiempo realizando actividades en el día a día, puede aportar información acerca de su salud cardiovascular.

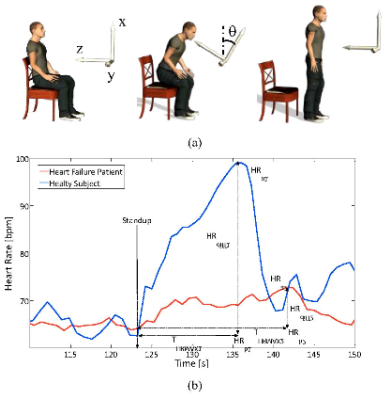


Ilustración 1 - imWell cambio en la actividad cardíaca introducida por la transición de estar sentado a de pie

Una vez los datos son capturados, son subidos al servidor mHealth.

Para la identificar un cambio postural comienzan con la detección de un cambio en el ángulo de la parte superior del cuerpo en relación con su posición recta y el cambio en la magnitud del vector de aceleración. Un cambio postural implica que dicho ángulo supere un cierto umbral.

Sin embargo, dicho ángulo puede que incluya imperfecciones producidas por un mal posicionamiento del aparato. Para eliminar dichas imperfecciones, se calcula la desviación estándar del ángulo en una ventana temporal de un segundo.

El sistema es capaz de detectar las transiciones posturales y como éstas afectan a un cambio en el rimo cardíaco. No obstante, no existe un estudio posterior que evalúe el estado cardiovascular y físico del sujeto.

1.1.2 Usando giroscopios y acelerómetros con sistema de rehabilitación tras una artroplastia total de rodilla[2]

Osteoartritis es la forma más común de artritis y artroplastia total de rodilla (de sus siglas en inglés TKA) es el tratamiento más común para los últimos escenarios de osteoartritis con un alto grado de éxito.

Sin embargo, la necesidad de una herramienta de monitorización comprensible tanto en casa como en el hospital era necesaria. Es por ello que se propuso en este estudio el uso de sensores para medir objetivamente la flexión/extensión del ángulo de la junta de la rodilla.

Para la monitorización se ha usado el “wearable” Opal que incluye en su interior acelerómetros y giroscopios triaxiales de APDM. Inicialmente fue probado en un brazo robótico, posteriormente en sujetos normales y finalmente en el análisis del paso y el ángulo de la rodilla. Asimismo, se visitaron algunas clínicas para entender el proceso de recuperación con datos objetivos.

Para llevar a cabo el proceso, dos sensores sin cables son colocados en la pierna, uno en el muslo y otro en la parte cercana al tobillo. Al colocar los sensores, pueden aparecer algunos errores ocasionados por un exceso de inclinación en los mismos. Para contrarrestar esta problemática, otros algoritmos tales como la rotación simple de matrices son usados para que el ángulo pueda ser medido con exactitud.

Cinco pacientes hicieron uno del sistema. La gráfica mostrada a continuación muestra su evolución a lo largo del tiempo.

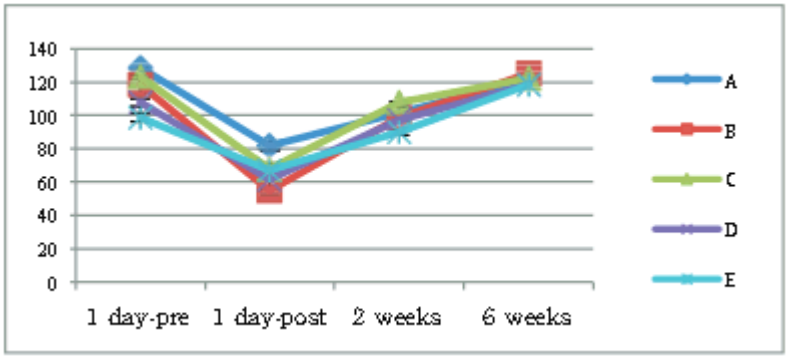


Ilustración 2 - Evolución cinco pacientes mediante el uso de Opal wearable

1.1.3 m-Physio[3]

m-Physio es una plataforma para realizar rehabilitación física personalizada basada en un acelerómetro. Especialistas y pacientes relacionados con el área de la rehabilitación pueden usar m-Physio para mejorar tanto la realización de ejercicios como su supervisión. Estas tareas no se suelen realizar de forma correcta por problemas de tiempo o por la distancia entre el paciente y el especialista encargado de su rehabilitación.

Este sistema hace uso de un móvil y un brazalete que capturan los datos relevantes de la rehabilitación del paciente. Una vez los datos en crudo son recogidos, un procedimiento de preprocesado filtra los datos del acelerómetro. Posteriormente se usa una técnica llamada Deformación Dinámica del Tiempo (de sus siglas en inglés Dynamic Time Warping) para entrenar y reconocer movimientos.

En la siguiente ilustración se puede apreciar los ajustes del acelerómetro, así como también las estadísticas recogidas del proceso de rehabilitación.

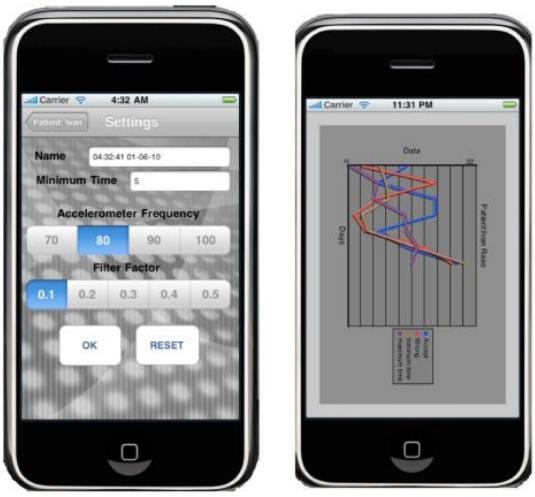


Ilustración 3 - m-Physio ajustes del acelerómetro y estadísticas recogidas del proceso de rehabilitación

Los datos en crudo producidos por el acelerómetro son ruidosos y redundantes, por lo que se usa la siguiente función de suavizado por cada eje:

$$S(A_t) = \begin{cases} A_t, & t = 0 \\ A_t * \alpha + S(A_{(t-a)} * (1.0 - \alpha)), & 0 < t \leq T \end{cases}$$

$S(A_t)$ es el vector filtrado de aceleración de salida y $A_t$es la aceleración cruda del vector de salida, el cual es adquirido mediante el dispositivo en el tiempo t. Además, α es el factor de suavizado cuyo dominio ∈ $[0,1]$. El factor α es crítico para que los datos sean válidos para su posterior análisis.

Para el reconocimiento de patrones, se ha usado el algoritmo DTW antes mencionado ya que requiere un entrenamiento simple y ha sido probado de forma efectiva en numerosos estudios. DTW computa la distancia entre dos ejercicios A y B encontrando el camino mínimo que será representado con un valor numérico. En esta aplicación, la distancia media euclidiana define el coste entre dos puntos distintos A_i y B_j de un ejercicio de rehabilitación.

Otro aspecto importante es la segmentación, en la cual se determina en comienzo y el final de un ejercicio. Esta parte es crucial, pues el paciente debe saber si está comenzando el ejercicio en la posición adecuada, así como también si el dispositivo debe saber cuándo ha terminado. Pese a que existen numerosas ecuaciones capaces de detectar el final de un ejercicio, la aplicación se basa en la última muestra para detectarlo. Seguiría el patrón siguiente:

1. El paciente pulsa el botón para comenzar el ejercicio.
2. El dispositivo comienza una cuenta atrás de cinco segundos para que el paciente pueda prepararse.
3. El dispositivo empieza a recoger los datos del paciente.
4. Finalmente, cuando el paciente finaliza el ejercicio, el dispositivo usa la última muestra para reconocer el final del mismo.

El método no sólo detecta el principio y el final del ejercicio, sino que también la posición en cada una de las fases.

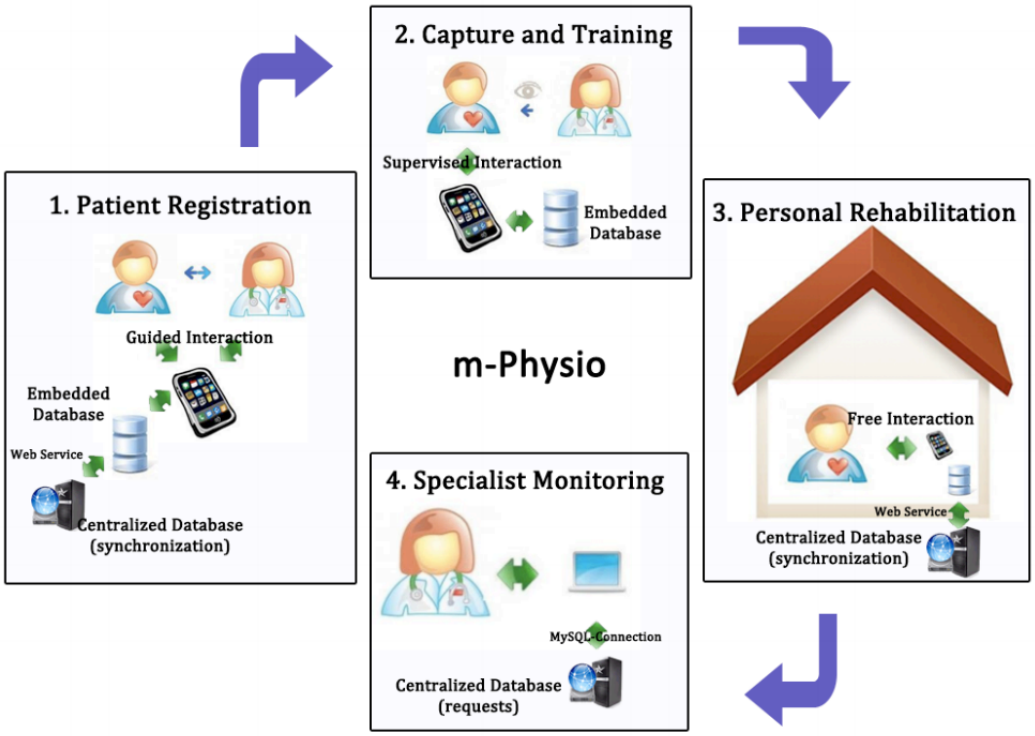


Ilustración 4 - Pasos principales de m-Physio

Gracias a m-Physio, los pacientes pueden llevar a cabo movimientos de rehabilitación sin la supervisión de un especialista, pero estando seguro la rigurosidad con la que lo realiza, obteniendo tras 15 días de rehabilitación un acierto de hasta el 93,3%.

1.1.4 Detección de deporte liviano basado en Smarwatches y Smartphones usando el k-ésimo vecino más cercano y el algoritmo DTW[4]

La implementación del reconocimiento de la actividad humana en la vida diaria es uno de los propósitos más buscados en el sector de la salud. Este estudio propone un sistema de detección de ejercicios de deporte liviano que pueden ser llevados a cabo por una persona. En estos ejercicios se encuentra caminar, flexiones o sentadillas. El sistema es capaz de reconocer el movimiento y de calcular el número de movimientos.

Para la extracción de los datos, se utilizan los datos de los acelerómetros procedentes tanto un Smartphone como de un smartwatch conjuntamente.

Posteriormente, se utiliza un sistema de ventana deslizante en el que se calcula y se segmenta la longitud del patrón en cada uno de los ejes. Tras esto, se conseguirán numerosas series de datos temporales desde un inicio a un final, en el que se encontrará el mejor valor gracias al algoritmo de Deformación Dinámica del Tiempo (DTW) tal y como se muestra en la siguiente ilustración:

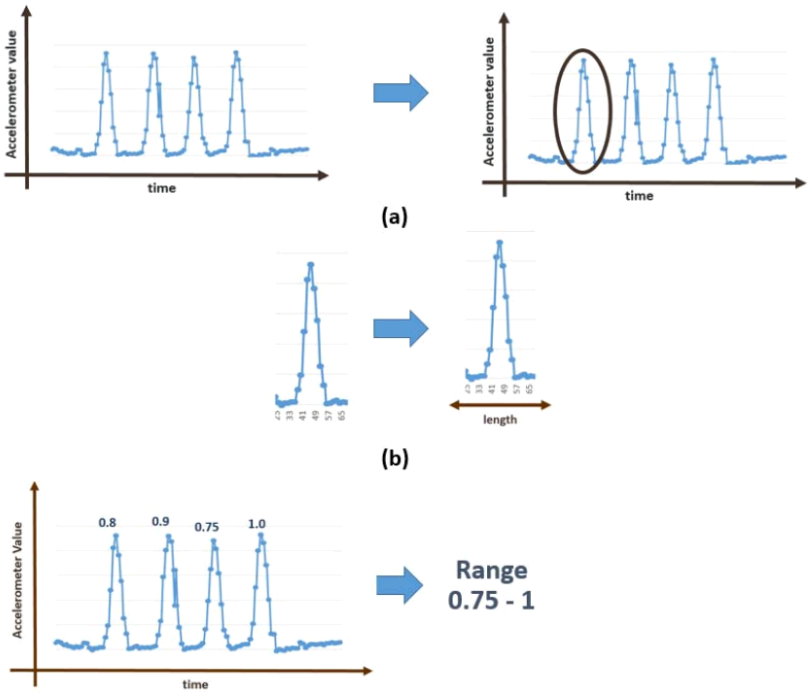


Ilustración 5 - Proceso de extracción de datos

Posteriormente, se realiza un preprocesado de los datos provenientes de los acelerómetros en los que se normaliza antes de ser clasificados mediante la siguiente función:

$$N = (d - d_{min}) + \left(\frac{N_{max} - N_{min}}{d_{min} - d_{max}}\right) + N_{min}$$

El símbolo N define datos normalizados, mientras que los datos que van a ser normalizados se definen por la letra d. Tras la aplicación de la fórmula, el rango de $N \in [1, -1]$

Una vez tenemos los datos normalizados, el estudio hace uso del algoritmo del k-ésimo vecino más cercano como algoritmo principal de clasificación. Tal y como se ha mencionado anteriormente, para el cálculo de la distancia de proximidad se ha utilizado el algoritmo DTW.

El sistema tiene algunos problemas tales como que el usuario debe realizar un retraso en la transición de un ejercicio a otro para que la probabilidad de error en el proceso de clasificación se reduzca. Además, el método de la ventana de tiempo deslizante hace uso de una cantidad ingente de recursos, lo que lo hace bastante impracticable.

1.1.5 e-FisioTrack[5]

Se trata de un entorno de telerehabilitación basado en el reconocimiento de movimientos mediante la captación de datos provenientes de un acelerómetro. El propósito general del estudio es mejorar la eficiencia de los procesos de rehabilitación gracias a los escenarios de los diversos pacientes externos. Además, está diseñado para acercar las posiciones de los pacientes y los fisioterapeutas, tratando de simular el caso ideal de tener un experto codo con codo con los pacientes al mismo tiempo que realizan los ejercicios prescritos en casa. Al mismo tiempo, el sistema permite a los fisioterapeutas conocer información exacta acerca de la evolución de sus pacientes.

Para llevar a cabo la tarea de monitorización de pacientes, se ha usado la arquitectura que se ilustra a continuación:

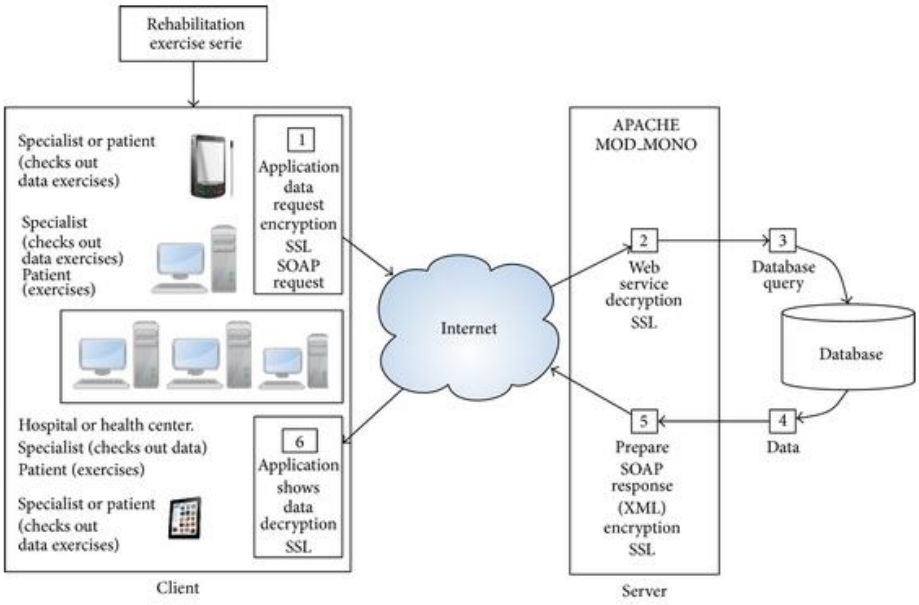


Ilustración 6 - Arquitectura eFisioTrack

La monitorización remota se lleva a cabo registrando los datos en crudo del acelerómetro. Una vez todos los datos han sido transmitidos a la aplicación, son filtrados usando la media entre tres puntos de movimiento para evitar valores que añadan ruido a la señal. Posteriormente, los valores filtrados del acelerómetro son la entrada del módulo de evaluación del movimiento. Este módulo hace uso del algoritmo Deformación Dinámica del Tiempo (DTW) capaz de evaluar si los gestos de los pacientes se están llevando a cabo correctamente. DTW compara con la repetición ideal y provee feedback tanto al paciente como al fisioterapeuta de cuántas repeticiones se han llevado a cabo, así como también la razón de cualquier repetición errónea que haya podido hacer.

1.1.6 Sistema de monitorización de ejercicios de peso libre[6]

Sistema capaz de reconocer de forma automática qué tipo de ejercicio es el que se está realizando gracias al uso de un acelerómetro triaxial. Dicho acelerómetro es colocado en un guante para poder monitorizar cada uno de los movimientos que se realizan con la mano.

El objetivo principal de este estudio es tanto detectar el tipo de ejercicio que se está realizando como el de contar el número de repeticiones que se han realizado.

El primer paso que debe realizar la aplicación es diferenciar si la persona se encuentra de pie o sentada, pues esto puede condicionar el reconocimiento del ejercicio.

Posteriormente, se realizaron varios ejercicios y se analizaron las gráficas de los datos que fueron captados:

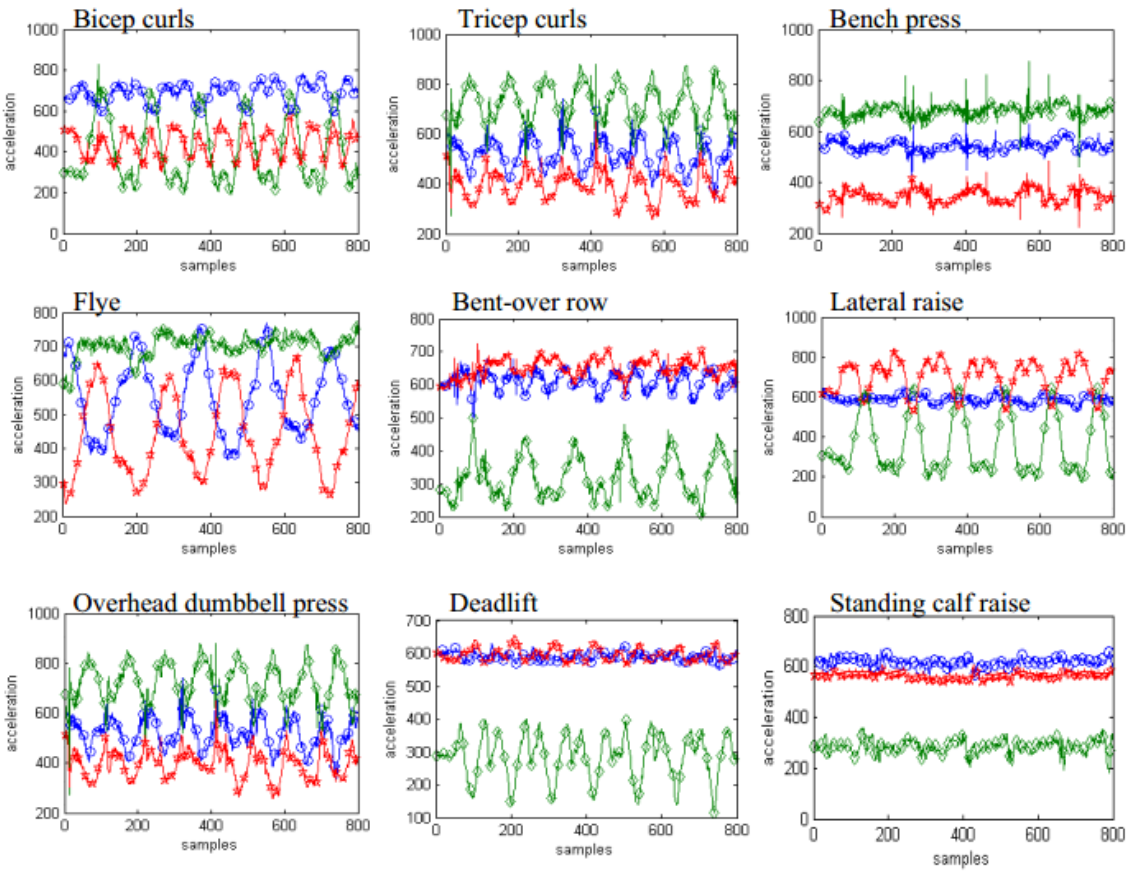


Ilustración 7 - Datos captados sistema de monitorización de ejercicios de peso libre

Tras el estudio de cada gráfica individual, se concluyó que existían cuantiosas diferencias entre cada uno de los ejercicios. Además, se estableció que se podía dilucidar en cada una de ellas la posición inicial, intermedia y final de cada uno de los ejercicios.

Para el reconocimiento del ejercicio se utilizó el Clasificador Naïve Bayes (NBC) junto con el Modelo Oculto de Markov (HMM).

Primero, se crea una ventana deslizante que solapa el 50% de las lecturas. Los valores de la ventana alimentan a NBC el cual ha sido previamente entrenado con vectores en una mezcla gaussiana. NBC clasifica cada ventana deslizante en un tipo de ejercicio concreto basado en la máxima probabilidad. HMM se utiliza como verificador. No obstante, en un instante dado, la cantidad de datos que puede ser suficiente para NBC, puede no serlo para HMM.

Una vez detectado el tipo de ejercicio, el sistema reconoce el patrón que se está repitiendo a lo largo del entrenamiento para contar cada una de las repeticiones.

1.2 Aplicaciones en el mercado

1.2.1 Freeletics

Freeletics es una aplicación que permite tener acceso a una cantidad ingente de rutinas disponibles en su plataforma. Actualmente se encuentra disponible tanto para Android como para iOS.

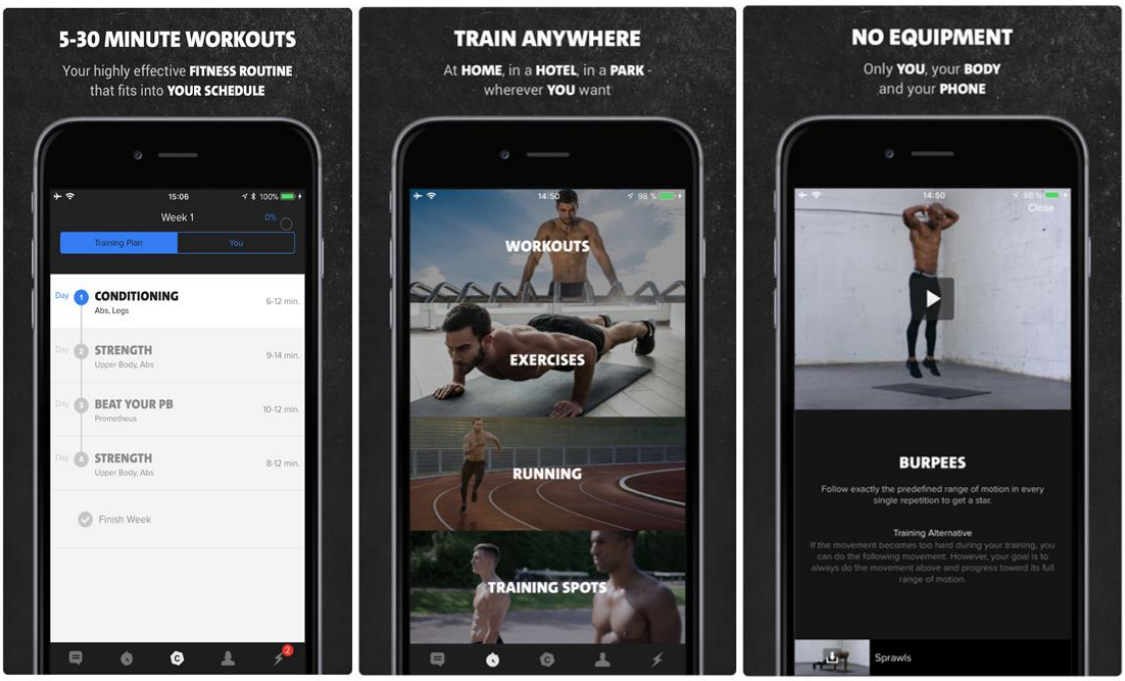


Ilustración 8 - App Freeletics

Freeletics comienza preguntando el estado físico actual de la persona y, en función de sus respuestas, se le asignaría una rutina con unos ejercicios de prueba. Tras finalizarla, el usuario recibirá más preguntas acerca de si le ha costado mucho, si la ha terminado, etc. Basándose en sus

respuestas, se le asignará una nueva rutina que vaya en concordancia con su estado físico actual. Este es el proceso que se utiliza para la recomendación de nuevas rutinas.

Asimismo, la aplicación posee un servicio Premium en el que existe la posibilidad de contratar un entrenador personal online, el cuál recomendará rutinas y ejercicios al usuario basándose tanto en sus respuestas como en su experiencia personal.

Puede acceder a la información completa a través de https://www.freeletics.com/es.

1.2.2 GetFit

GetFit es una aplicación que posee una base de datos con una gran variedad de rutinas. Para poder acceder a ella se debe pagar una suscripción mensual.

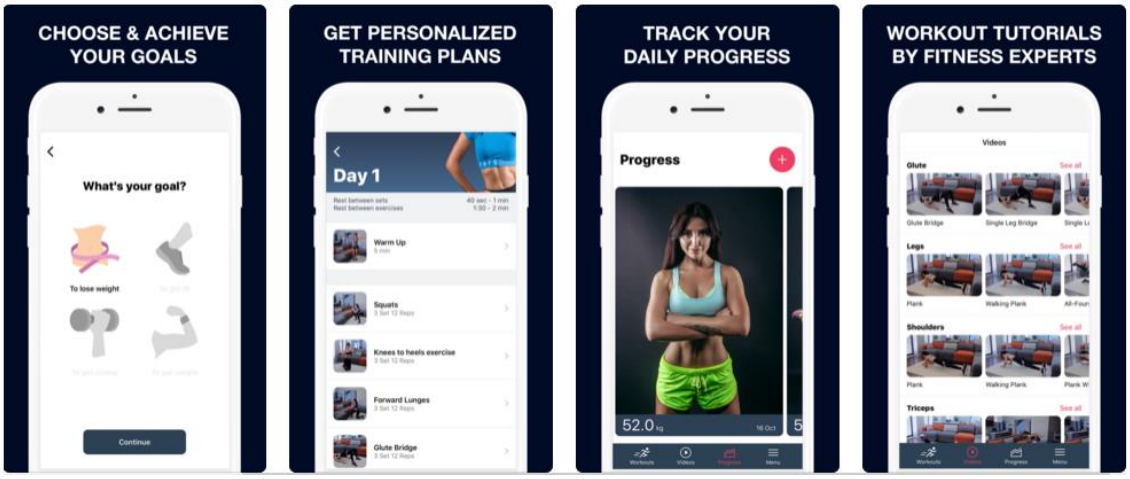


Ilustración 9 – Aplicación GetFit

Por cada uno de los ejercicios de la rutina aparece un vídeo explicativo de cómo se debe realizar. Tras terminar cada ejercicio, la aplicación pregunta al usuario el número de repeticiones que ha realizado. Una vez termina todos los días que la rutina estipula, se le asignará la siguiente en función de su rendimiento y objetivo.

Por último, muestra al usuario su progresión en forma de gráficas.

Puede acceder a la información completa a través de https://itunes.apple.com/us/app/getfit-home-fitness-workout/id1273749004?mt=8.

1.2.3 Runtastic

Runtastic es una aplicación muy completa que posee numerosas rutinas de diferentes niveles. Se centra en la monitorización de sesiones de footing, calculando la distancia recorrida, el tiempo que se ha tardado, así como también el ritmo cardíaco.

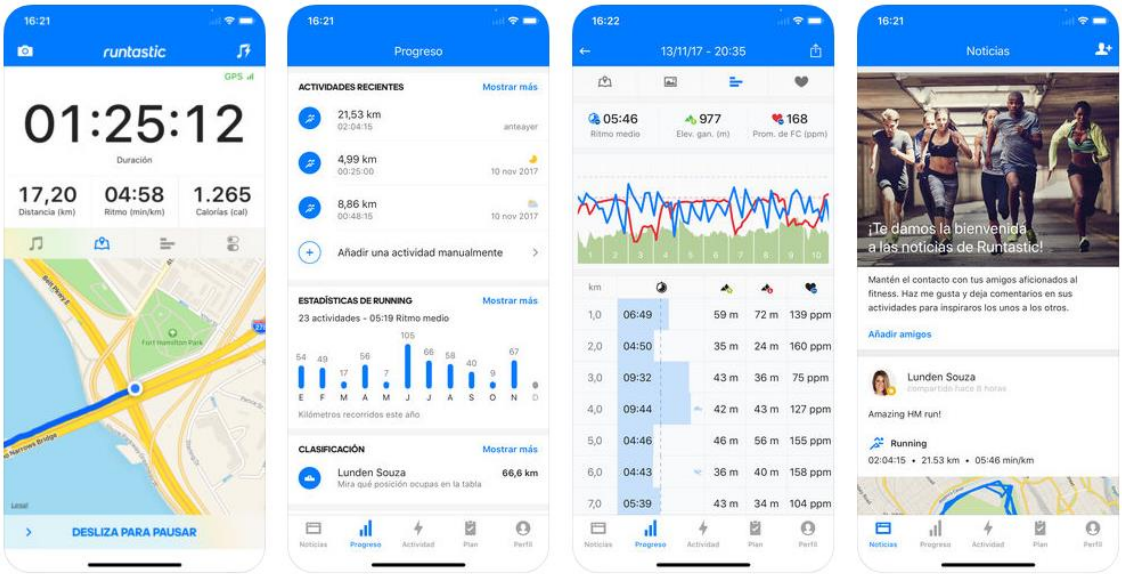


Ilustración 10 - App Runtastic

Además de monitorizar tu actividad, posee un entrenador personal por voz, el cual da feedback personalizable al usuario, pudiendo elegir entre velocidad, distancia, tiempo, etc.

Asimismo, en caso de hacer deporte con amigos, registraría quién de todos es el que más distancia ha recorrido, recibiendo mensajes de apoyo para mantener al usuario motivado.

Runtastic es en sí una aplicación para realizar el seguimiento de actividades como correr, senderismo e incluso esquí. No obstante, esta misma empresa posee numerosas sub aplicaciones. Un subconjunto de ellas se muestra en la ilustración mostrada a continuación:

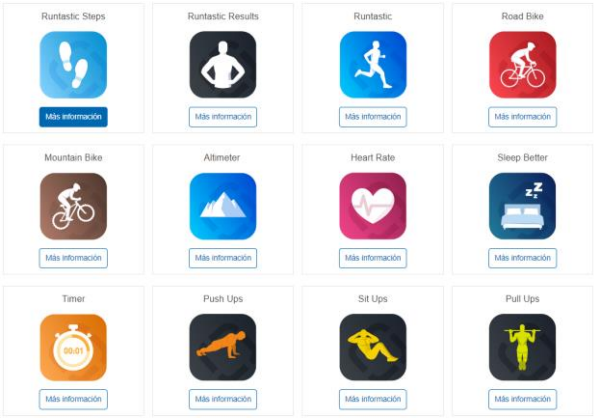


Ilustración 11 - Sub aplicaciones Runtastic

Nos hemos centrado en el estudio de aquellas que son de relevancia para IASport.

1.2.3.1 Push Ups

Push Ups es una aplicación que, tras unas preguntas tales como si se está en forma o cuántas repeticiones le apetece hacer al usuario, comienza la monitorización del ejercicio en función de dichas respuestas. La forma que tiene de contar cada una de las repeticiones es muy básica. La aplicación explica al usuario que debe colocar su móvil en el suelo y, cada vez que presione la pantalla con su nariz, contará como una repetición satisfactoria.

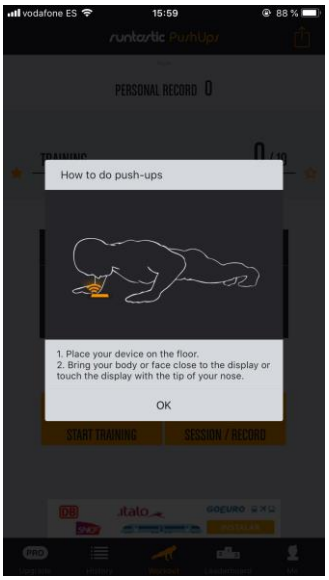


Ilustración 12 - Runtastic Push up colocación móvil

Tras la finalización de la serie, la aplicación permite la posibilidad de guardar el número de repeticiones llevadas a cabo para futuras estadísticas.

La aplicación no es capaz de detectar errores en el proceso de la repetición, así como tampoco errores en los tiempos de la repetición.

1.2.3.2 Sit Ups, Pull Ups y Squats

Sit Ups, Pull Ups y Squats son tres aplicaciones de la rama de Runtastic que son las únicas que hacen uso del acelerómetro para contar repeticiones. Se basa en controlar si el valor límite del acelerómetro supera o no a un valor máximo por defecto. Sólo se controla si se han realizado todas las repeticiones, pero es incapaz de detectar que los tiempos se hayan seguido rigurosamente o si la técnica del ejercicio ha sido la correcta.

Tras terminar el ejercicio, al igual que la aplicación de Runtastic Push Ups, te permite almacenarlas para futuras estadísticas y evaluar tu progresión.

Para que las aplicaciones sean capaces de detectar correctamente el número de repeticiones realizadas, el móvil debe ser colocado en una posición concreta. Para ello, antes del ejercicio se muestra una breve explicación tal y como la que a continuación se ilustra:

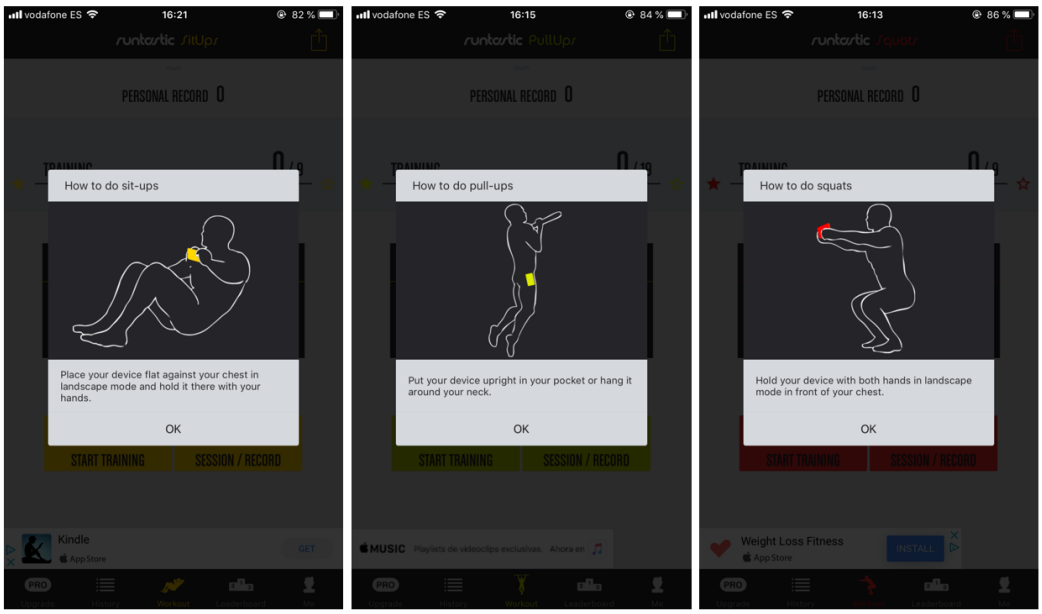


Ilustración 13 - Posición móvil Apps Sit Ups, Pull Ups y Squats de Runtastic

Para conocer más información acerca de esta o cualquier otra aplicación de Runtastic, consulte el enlace https://www.runtastic.com.

1.2.4 8Fit

8Fit es una aplicación que permite aunar rutinas y recetas en un mismo lugar. Posee elenco de rutinas a las que poder apuntarse en función del objetivo que se desee conseguir. Asimismo, a medida que se van completando las rutinas, la aplicación da por sentado que se ha completado en su totalidad y te apunta a otra rutina superior pero siempre manteniendo el objetivo.

Asimismo, las rutinas vienen acompañadas por hábitos y dietas que ayudan a conseguir el objetivo deseado.

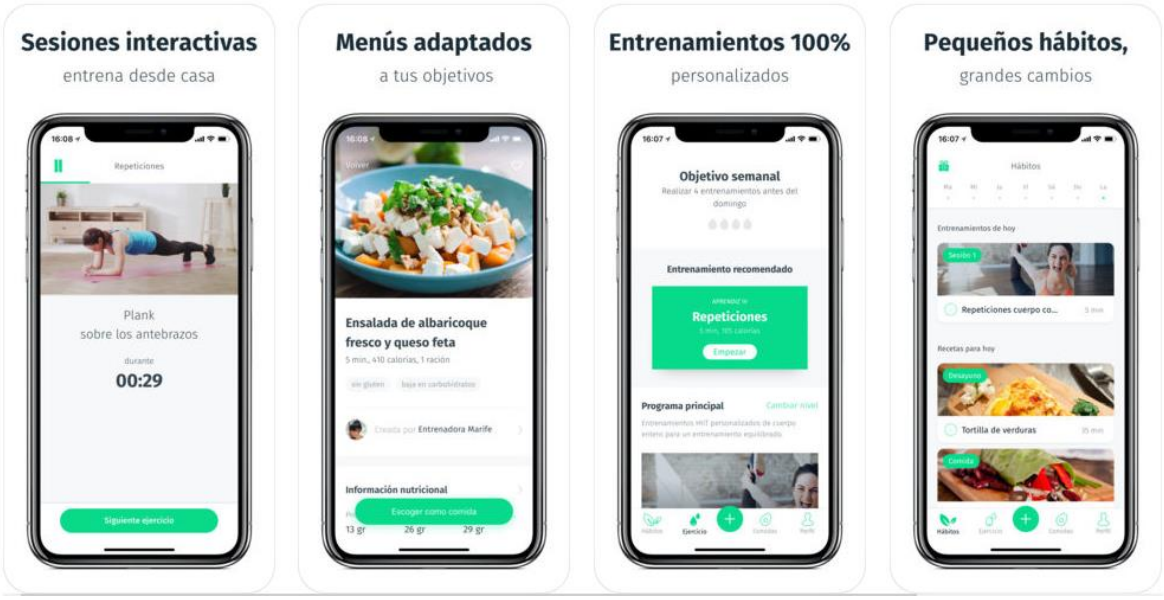


Ilustración 14 - Capturas pantalla 8fit

Puede acceder a la información completa a través de https://8fit.com/es/.

2. Objetivos

Habiendo analizado tanto aplicaciones como estudios similares del mercado actual y expuesto el problema al que nos enfrentamos, en este punto se establecerán los objetivos a conseguir por el proyecto.

2.1 API REST

La API REST, será la piedra angular a partir de la cual se articulará todo el proyecto.

En primer lugar, proveerá al proyecto de una independencia entre backend y frontend. Cualquier cliente podrá consumir los recursos que precise de nuestra API, ofreciendo la flexibilidad y escalabilidad que todo proyecto necesita.

La API permitirá realizar todas las acciones del sistema, permitiendo registrar y autenticar a los usuarios que harán uso de la API, permitiéndoles crear ejercicios, crear tipos de entrenamiento, crear rutinas, apuntarse a rutinas, recibir estadísticas personalizadas y un largo etcétera.

2.2 CDN

La CDN se utilizará como un refuerzo a la API, eliminando de ésta todas las peticiones referentes a la obtención de recursos de tipo imagen. Esta eliminación la liberará de peticiones con una carga significativa.

En la aplicación se necesitan estos tipos de recursos para, por ejemplo, mostrar la manera en la que el ejercicio debe realizarse, ya que esta información es mostrada en forma de gif.

Asimismo, la posesión de una CDN hace que el posicionamiento SEO se vea beneficiado, ya que está ligando indirectamente a éste.

2.3 Cliente móvil iOS

Nuestro cliente móvil para iOS posee dos grandes objetivos.

Por una parte, la aplicación consumirá los recursos de la API REST previamente mencionada, permitiendo a los usuarios logarse/registrarse, apuntarse a una rutina, listar las rutinas a las que está apuntado, ver sus estadísticas, entrenar, etc..

Por otra parte, es donde se encuentra el algoritmo capaz de detectar si se está llevando a cabo correctamente una serie de un ejercicio mediante el análisis de datos en tipo real. Tras la finalización del entrenamiento, la aplicación enviará un informe con el número de repeticiones correctas/incorrectas y, en caso de que haya incorrectas, qué tipo de error es el que ha cometido.

2.4 Web

Por último, una web que será el frontend responsive de la administración de la plataforma.

La web consumirá los recursos necesarios de la API, permitiendo a los usuarios visualizar los detalles de sus rutinas, de las que no están apuntados, apuntarse a nuevas rutinas, ver sus estadísticas, etc. Asimismo, le permitirá a los trainers crear rutinas, ejercicios, tipos de entrenamiento, entrenamientos personalizados, editar todos los parámetros que considere necesario, etc.

3. Tecnologías

Las tecnologías han sido minuciosamente seleccionadas para hacer frente a los requerimientos del proyecto, permitiendo la flexibilidad, escalabilidad y mantenibilidad que éste requiere.

3.1 Play Framework

Play Framework es un framework de aplicaciones web de alta productividad en Java y Scala que integra componentes para el desarrollo de aplicaciones web modernas.

Play está basado en una arquitectura ligera, sin estado y amigable. Además, posee un consumo mínimo de recursos (CPU, memoria e hilos), permitiendo así aplicaciones completamente escalables gracias a su modelo basado en Akka Streams.

Ilustración 15 - Play framework

Play Framework permite realizar API's REST con total comodidad tanto en Java como Scala.

Para esta aplicación, se ha hecho uso de su versión en Java con la que se ha construido la API REST encargada de la administración de rutinas, ejercicios, tipos de entrenamiento, estadísticas y usuarios.

Sigue una arquitectura MVC (Modelo, Vista, Controlador). No obstante, la utilización de las vistas es completamente opcional, permitiéndonos devolver directamente JSON que es consumido directamente por los dos actuales clientes (Web Laravel e iOS Swift).

3.2 Laravel

Laravel es un framework MVC de código abierto para PHP muy potente con el que construir proyectos web.

Se caracteriza por su sistema de rutas RESTful, su gestión de plantillas por medio de “blade” y el uso del ORM “Eloquent”. Asimismo, utiliza numerosos componentes de Symfony.

En IASport, se ha usado para construir el cliente web. En nuestro caso, se ha utilizado el controlador como consumidor de la API en Play Framework, así como también para controlar la lógica de las diferentes vistas. Su gestión de plantillas y su sencillez son los que hacen a Laravel de un framework idóneo.

Ilustración 16 - Laravel Logo

3.3 MySQL

MySQL es el sistema de gestión de bases de datos relacionales más popular. Fue desarrollado y distribuido por Oracle Corporation.

Ilustración 17 - MySQL Logo

MySQL se ha utilizado como motor de bases de datos para dotar a IASport de persistencia.

3.4 Swift

Swift es el lenguaje multiparadigma creado por Apple con el objetivo de programar software para móviles, ordenadores de escritorio, así como también para servidores. Está diseñado para integrarse con los Frameworks Cocoa y Cocoa Touch, pudiendo hacer uso de bibliotecas en Objective-C y realizar llamadas a C. Desde el 2015 se considera código abierto.

Ilustración 18 - Swift Logo

En el proyecto se ha utilizado para programar el cliente en iOS para iphone's.

3.5 Node.js

Node.js es un entorno de ejecución para JavaScript construido con el motor V8 de Google. Haceuso de un modelo de operaciones E/S sin bloqueo y es orientado a eventos.

Ilustración 19 - Node.js Logo

En el proyecto se ha hecho uso de Node.js para construir la CDN encargada de servir los recursos de tipo imagen a los diferentes clientes, aliviando el exceso de tráfico que generan este tipo de solicitudes. Nuestro servidor en Node.js se encargaría de servirlas, eliminando la responsabilidad a nuestra API principal, haciendo que el servicio funcione de forma mucho más eficiente.

3.6 JPA

JPA (Java Persistence API) es un framework de Java encargado de manejar datos relacionales en aplicaciones usando la plataforma Java SE y Java EE. El objetivo principal de JPA es aprovechar las ventajas de los lenguajes orientados a objetos al interactuar con una base de datos (siguiendo el patrón ORM (Object Relational-Mapping), permitiendo así seguir utilizando objetos regulares conocidos como POJOS.

En IASport se ha usado en la API desarrollada con Play Framework. Gracias a este framework, el acceso a la base de datos se encuentra completamente abstraído, pudiendo realizar cualquier acción CRUD de una manera directa. JPA, además, aporta al proyecto flexibilidad, pues permite añadir o eliminar atributos de las diferentes entidades sin tener que realizar prácticamente cambio alguno en el resto de las capas.

4. Metodologías

Las metodologías del desarrollo han sido seleccionadas para garantizar que el proyecto se desarrolla de una manera rigurosa, adaptándose a los cambios y con éxito. Se ha optado por una metodología ágil de desarrollo tales como Scrum y Kanban, adaptadas a un solo miembro.

4.1 Scrum y Kanban

Para el desarrollo del proyecto se han combinado las técnicas de Scrum y Kanban. Se ha seguido un diseño iterativo, realizando sprints con una duración de tres semanas.

Al final de cada sprint se realizaba una reunión con el tutor. En dicha reunión se hacía una pequeña demo de las nuevas funcionalidades incorporadas, así como también una retrospectiva de problemas que hayan podido surgir. Asimismo, se planificaban los nuevos objetivos que iban a ser abordados en el siguiente sprint.

Siguiendo la filosofía Kanban, se ha organizado el proyecto en tarjetas sobre las que se aplican una serie de reglas específicas. Entre esas reglas se encuentran la de evitar que un bug sea trasladado al siguiente proceso o la de “stop starting, start finishing”, heredada de la filosofía Lean

4.2 Flujo de trabajo: GitFlow

El flujo de trabajo utilizado es GitFlow, modelo de Git basado en ramas que se ajusta perfectamente a las filosofías de Scrum y Kanban, permitiendo un diseño iterativo.

Las ramas principales son:

- master: rama donde el equipo de desarrollo integra el código para la siguiente versión del proyecto.
- production: rama que contiene el código que debe estar en producción.

Los tres subgrupos de ramas auxiliares son:

- feature/name: rama en la que se desarrollan nuevas características de la aplicación. En nuestro caso reciben el nombre de TIC-x, siendo x el número de ticket que lo identifica en el tablero Trello. Una vez terminadas y testeadas, son incorporadas a la rama master.

- hotfix/name: las ramas hotfix se realizaban de master a master.
- release/versión: rama donde se prepara el código y se corrigen los últimos fallos antes de que pase a producción.

4.3 Control de versiones: GitHub

Para el control de versiones se ha utilizado GitHub, la plataforma de desarrollo colaborativo que utiliza el CVS Git. Es idóneo para el proyecto IASport, permitiéndonos de una manera cómoda mantener ordenado todo el desarrollo, realizar merges, pull requests, etc.

4.3.1 Repositorios

Para el desarrollo de IASport hemos hecho uso de dos repositorios en GitHub:

- pcl23-tfg-backend: Repositorio en el que se encuentra todo el código pertinente de la API (Play Framework), web (Laravel), así como también la CDN (Node.js)
- pcl23-tfg-ios: Repositorio en el que se encuentra el código referente a la aplicación móvil en Swift para iPhone.

4.3.2 Pull Requests

Cada vez que una tarea se termina, se crea un pull request asociada a la misma. En caso de que se acepten los cambios a incorporar (en este caso por el único miembro), el Pull Request sería mergeado.

Tal y como se detallará a continuación, cada tarjeta (ticket) debe tener asociado un pull request, el cual será añadido como comentario.

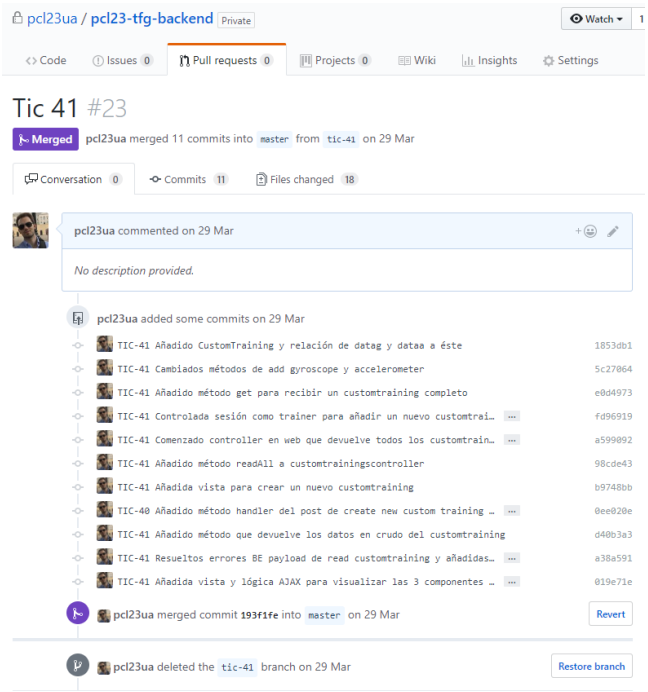


Ilustración 20 - Ejemplo Pull Request

Al finalizar el proyecto, se obtuvieron las siguientes estadísticas por cada repositorio:

- pcl23-tfg-backend:
  - 41 pull requests.
  - 280 commits.
  - 1 release
- pcl23-tfg-ios:
  - 34 pull requests.
  - 344 commits.
  - 1 release

4.4 Trello

Trello ha sido la herramienta principal de gestión de historias de usuario, tickets, funcionalidades y bugs utilizada en este proyecto a modo de tablero virtual de Kanban.

En él se han incorporado numerosas listas en el que todas estas tareas han sido agrupadas. Cada tarea está representada por una tarjeta.

4.4.1 Tarjetas

Cada una de las tarjetas de Trello representa una tarea del proyecto. El nombre de la misma da información sobre qué aportará al proyecto.

Cada una de las tareas tiene asignada una o varias etiquetas (labels) de un color determinado para poder determinar la naturaleza de la misma, así como también el sprint en el que se ha realizado. La lista de labels ha sido la siguiente:

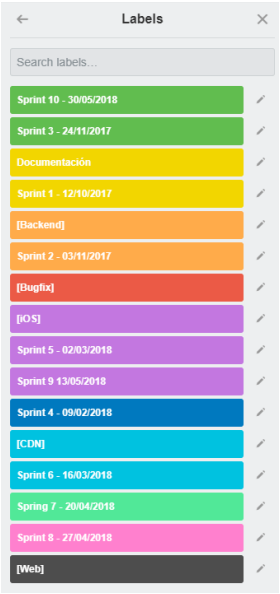


Ilustración 21 - Lista labels Trello

Las tarjetas pueden o no ser tickets. Un ticket implica que debe haber una adición o modificación de código una vez haya finalizado y van asignadas a un pull request. Además, el nombre debe empezar por TIC-X, siendo X el identificador del ticket en cuestión.

Cada una de las tarjetas terminadas debe tener en la parte de los comentarios el enlace al pull request al que esté asociado (en caso de que sea un ticket) o al documento o recurso (en caso de que no sea un ticket).

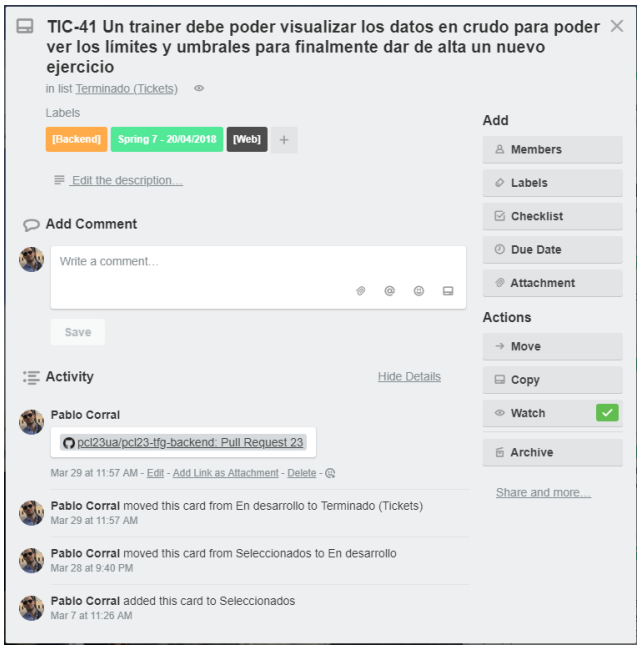


Ilustración 22 - Ejemplo tarjeta tipo ticket Trello

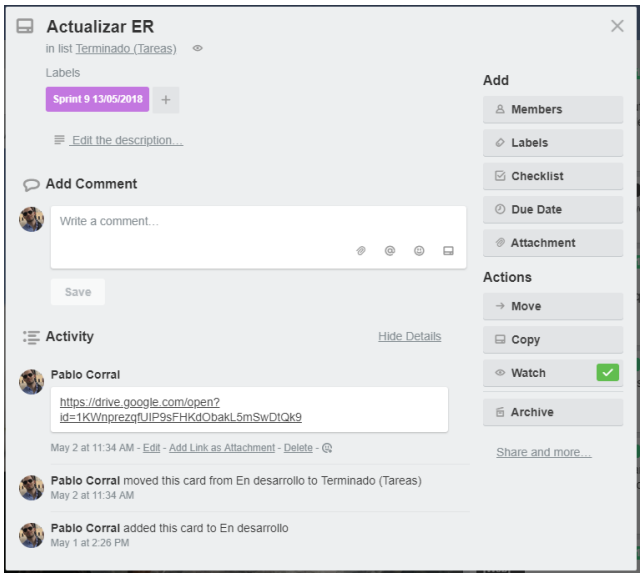


Ilustración 23 - Ejemplo tarjeta normal Trello

4.4.2 Listas

Las listas principales utilizadas son:

- Backlog: Tareas del proyecto que se encuentran todavía por desarrollar y que no van a ser desarrolladas en el sprint actual.
- Seleccionados: Conjunto de tareas que serán desarrolladas en el sprint actual.
- En desarrollo: Tareas que se están llevando a cabo.
- Testing: Conjunto de tareas que estén siendo testeadas.
- Terminado (Tickets): Conjunto de tareas que se hayan terminado que sean del tipo ticket.
- Terminado (Normal): Conjunto de tareas terminadas y que no sean del tipo ticket (p. ej. documentación).

Después de cada reunión de retrospectiva, se realiza el análisis de las nuevas funcionalidades a desarrollar y de los bugs que deben ser eliminados. Una vez el proceso de análisis haya finalizado, se divide el trabajo en pequeñas tareas (tarjeta de trello) y se añaden a “Backlog”. Posteriormente, se pasan a “Seleccionados” aquellas que vayan a ser llevadas a cabo en el sprint.

Una vez se quiera comenzar una tarea, ésta pasará a “En desarrollo”. Una vez finalizada pasará a “Testing” en caso de que sea un ticket, donde se le realizarán pruebas para verificar su correcto funcionamiento y finalmente a “Terminado (Tickets)”. En caso de que no sea un ticket, pasará a “Terminado (Normal)”.

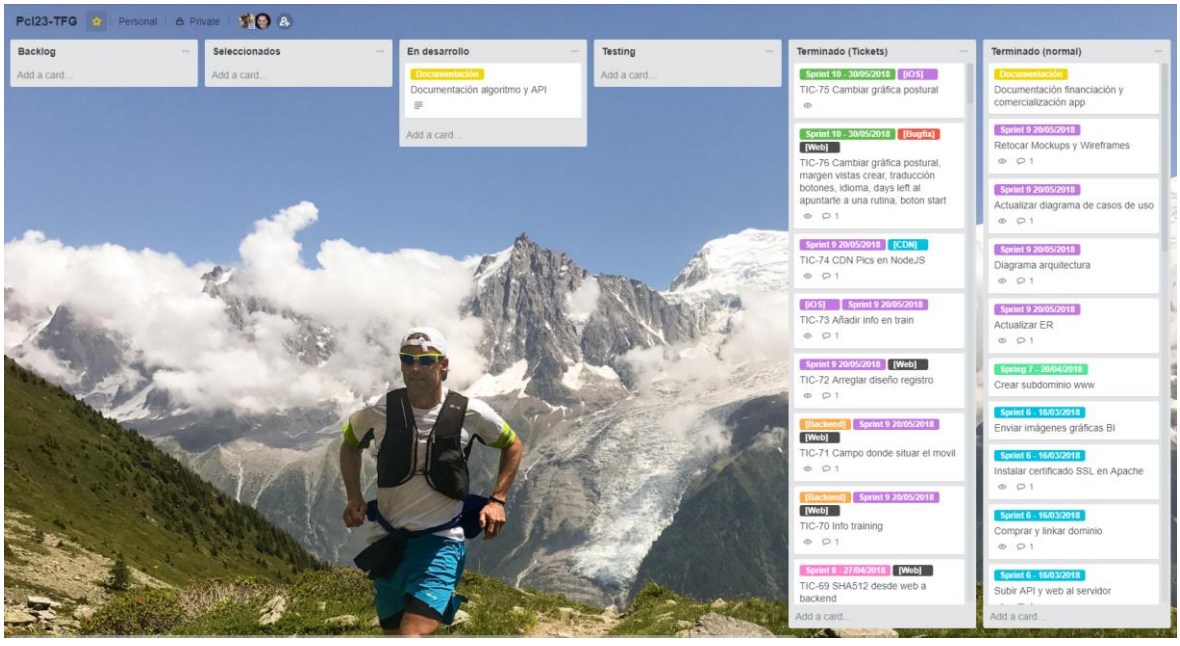


Ilustración 24 – Tablero Trello

Tras los diez sprints del proyecto, se generaron 104 tarjetas de Trello, 75 de ellas tickets.

4.5 Releases

Se ha realizado un release por cada uno de los componentes que forman IASport. Estos contienen todas las funcionalidades descritas en el presente documento. Para acceder a la versión, puede hacerlo haciendo clic en el link de la misma que desee visitar. A continuación, se describirán brevemente.

4.5.1 API

- Versión v1.0.0
    - Registro y autenticación de usuarios
    - Administración de rutinas, ejercicios y tipos de entrenamiento
    - Gestión Custom trainings
    - Estadísticas
    - Gestión de informes e histórico de entrenamientos

4.5.2 Cliente móvil iOS

- Versión v1.0.0
    - Registro y autenticación
    - Entrenar custom training
    - Visualización estadísticas
    - Entrenar serie
    - Sistema de monitorización de entrenamiento

4.5.3 Web

- Versión v1.0.0
    - Registro y autenticación
    - Administración de rutinas, ejercicios y tipos de entrenamiento
    - Gestión custom trainings
    - Gestión rutinas

- Visualización estadísticas

4.5.4 CDN

- Versión v1.0.0:
  - Subir recurso
  - Servir recurso

5. Arquitectura e implementación

La arquitectura de IASport debe poder hacer frente a todos y cada uno de los requerimientos, comenzando con la API, siguiendo con la web y terminando con el cliente móvil iOS.

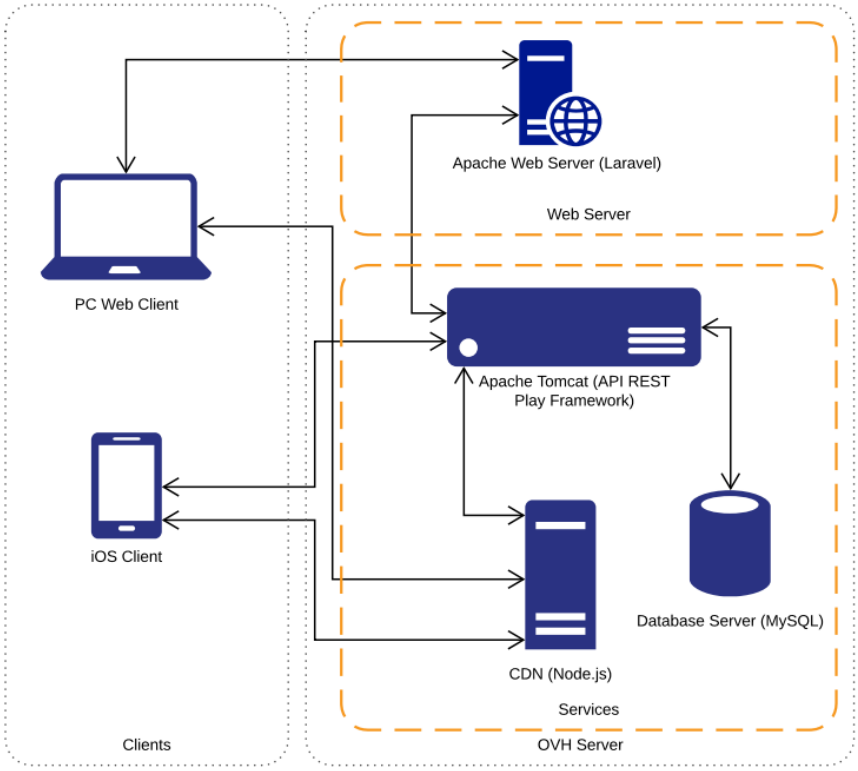


Ilustración 25 - Arquitectura IASport

La API REST es la piedra angular a partir de la cual se articula el sistema en su totalidad. Ésta es consumida por el resto de clientes (iOS y Web). Las acciones que se pueden llevar a cabo se encuentran restringidas al rol del usuario logado.

En IASport existen tres roles de usuario:

- Usuario no autenticado: usuario que puede registrarse y autenticarse en el sistema.
- Usuario normal: usuario que puede gestionar rutinas, visualizar sus estadísticas y entrenar.
- Trainer: usuario encargado de la administración de la plataforma, pudiendo administrar rutinas, ejercicios, tipos de entrenamiento, así como también crear custom trainings.

Cabe destacar que el usuario normal puede realizar las acciones del usuario no autenticado y que el trainer puede realizar las acciones de un usuario normal. Para una mayor comprensión véase el diagrama de casos de uso del anexo.

A continuación, se muestran las diferentes responsabilidades de cada una de las partes de la arquitectura. Los colores blanco, azul y naranja representan las acciones que pueden ser realizadas por un usuario no identificado, un usuario normal y un trainer respectivamente:

Ilustración 26 - Detalle arquitectura IASport

5.1 Modelo de datos

El modelo de datos debe poder dotar de persistencia de toda la aplicación, albergando todos y cada uno de los parámetros necesarios de cada una de las entidades.

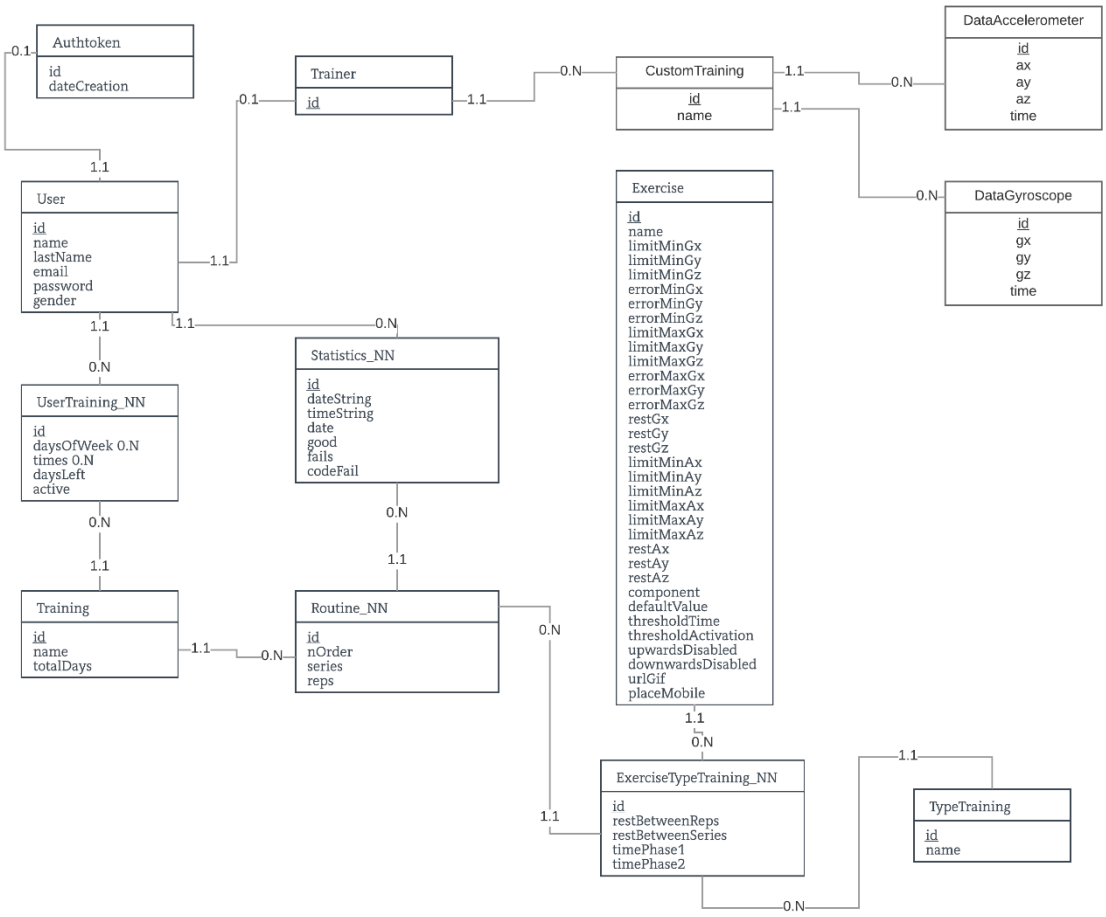


Ilustración 27 - Entidad Relación

Cabe destacar que JPA no permite añadir parámetros en una relación muchos a muchos de una manera completamente rigurosa, por lo que se ha optado por hacer una entidad intermedia (nombradas como X_NN en el diagrama), y que poseen una relación 1.N con cada una de las dos entidades que relaciona.

5.2 API REST

La API REST es la pieza principal del sistema. Presenta las siguientes capas:

- Modelo: capa encargada de realizar el acceso a la capa de datos mediante el patrón DAO (Data Access Object). El resto de la aplicación quedaría aislada de todo lo que conlleve el acceso a la base de datos, manteniendo un bajo acoplamiento y alta cohesión. Esta capa contiene todos

los modelos de IASport. Cada modelo representa una entidad en la base de datos (Exercise, Typetrainign, etc). Para ver el listado completo de entidades, puede ver el modelo de datos.

- Servicio: capa que contiene la lógica de negocio de la aplicación, aislándola tanto del modelo como del controlador.
- Controlador: última capa donde se reciben las peticiones y se generan las respuestas pertinentes. Utiliza la capa de servicio para obtener dichas respuestas.

Una capa “Vista” no es necesaria en la API, pues todos los datos a enviar son “parseados” en JSON.

Un ejemplo representativo de la API REST, es la solicitud de todos los parámetros que el algoritmo necesita para llevar a cabo la monitorización del ejercicio.

Realizamos una petición GET a la URL /exercisetypetraining/7 (lugar donde se encuentra nuestra API REST) con la cabecera Authorization junto con el token. La respuesta recibida es la siguiente:

```
{ ▼ 2 properties, 994 bytes
  "code": 200,
  "exerciseTypeTraining": { ▼ 7 properties, 958 bytes
    "id": 7,
    "exercise": { ▼ 34 properties, 812 bytes
      "id": 8,
      "name": "Squat",
      "limitMinGx": -0.5,
      "limitMinGy": 0.85,
      "limitMinGz": -0.7,
      "errorMinGx": "Do not rise your heels",
      "errorMinGy": "Get down straight",
      "errorMinGz": "Do not bend your knees more than 90º",
      "limitMaxGx": 0.5,
      "limitMaxGy": 1.0,
      "limitMaxGz": 2.5,
      "errorMaxGx": "Do not bend forward",
      "errorMaxGy": "Do not bend sideways",
      "errorMaxGz": "Do not bend backwards",
      "restGx": 0.0,
      "restGy": -0.85,
      "restGz": -1.25,
      "limitMinAx": -0.25,
      "limitMinAy": -1.85,
      "limitMinAz": 1.25,
      "limitMaxAx": -0.25,
      "limitMaxAy": -1.25,
      "limitMaxAz": 1.25,
      "restAx": 1.0,
      "restAy": -1.0,
      "restAz": -0.5,
      "component": "ax",
      "defaultValue": -1.0,
      "thresholdTime": 0.5,
      "thresholdActivation": 0.25,
      "upwardsDisabled": true,
      "downwardsDisabled": true,
      "urlGif": https://iasport.es:3000/cdn/resource/178399047191,
      "placeMobile": "Calf"
    },
    "typeTraining": { ▼ 2 properties, 29 bytes
      "id": 2,
      "name": "Hipertrofia"
    },
    "restBetweenReps": 0.0,
    "restBetweenSeries": 60.0,
    "timePhase1": 3.0,
    "timePhase2": 2.0
  }
}
```


Ilustración 28 - Ejemplo JSON API REST

El listado completo de métodos de la API REST se encuentra en el anexo.

5.3 Web

La web es la encargada de la administración completa del servicio. Permite a los usuarios registrarse, logarse, visualizar sus estadísticas, y a los “trainers” crear ejercicios, rutinas, tipos de ejercicios, etc.

Posee las siguientes capas:

- Controlador: capa encargada de realizar las peticiones a la API, estructurar las respuestas, controlar la lógica de las vistas y finalmente servirlas. Esta capa consume la API REST para llevar a cabo todas las acciones.

- Vista: capa que se encarga de la visualización de la interfaz de usuario. Posee un flujo de comunicación con el controlador para el envío/recepción de datos.

5.4 Cliente móvil iOS

El cliente móvil ha sido desarrollado para iOS en Swift.

Posee las siguientes capas:

- Controlador: capa encargada de la lógica de la aplicación y de la vista. Asimismo, realiza las peticiones pertinentes a la API.
- Vista: capa que se encarga de la visualización de la interfaz de usuario. Posee un flujo de comunicación con el controlador para el envío/recepción de datos.

El cliente móvil iOS es donde se encuentra el sistema de monitorización de ejercicios.

Además de entrenar, tiene las siguientes funcionalidades:

- Como usuario normal:
    - Gestionar rutinas
    - Visualizar estadísticas (perfil)
- Como trainer:
    - Capturar datos para un custom training

5.5 CDN

La CDN, realizada en Node.js, es la encargada de servir recursos de tipo imagen. La API principal almacena una referencia a cada uno de los recursos para que los clientes puedan consultarlos.

Los dos métodos que posee son:

- POST /upload: método encargado de añadir una imagen nueva a la CDN.
- GET /public/{name}: método encargado de servir la imagen que tenga por nombre “name”.

Actualmente, las únicas imágenes que posee IASport son las de cómo realizar un ejercicio. No obstante, la CDN está preparada para servir cualquier imagen.

6. Sistema de monitorización de ejercicios

6.1 Análisis de datos provenientes de los sensores del iPhone

El primer paso para diseñar nuestro sistema de monitorización fue estudiar los datos en crudo provenientes del acelerómetro y giroscopio del iPhone. Para ello, se decidió crear una aplicación y una API destinadas al envío y captación de estos datos para su posterior visualización.

Se realizaron doce repeticiones de flexiones de pecho con el dispositivo en el bíceps derecho, tardando tres segundos en bajar y uno en subir.

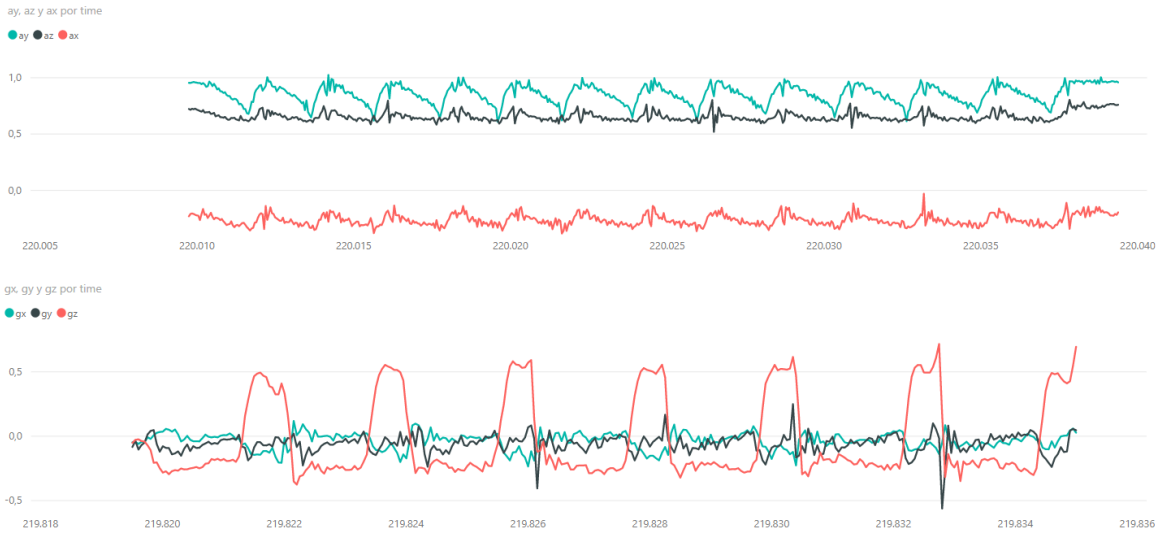


Ilustración 29 - Datos en crudo doce repeticiones de flexiones de pecho visualizadas con Power BI

En la primera gráfica se muestran las tres componentes del acelerómetro (eje y) frente al tiempo en milisegundos (eje x).

En la segunda, se muestran las tres componentes del giroscopio (eje y) frente al tiempo en milisegundos (eje x).

Lo primero que se puede apreciar en cada uno de los componentes es la vibración que se introduce en el ejercicio por parte del deportista. Claramente, esto debía ser el primer problema a resolver.

Asimismo, y tras estudiar la salida de numerosos ejercicios, pudimos dilucidar que existe una componente principal para el acelerómetro y giroscopio, es decir, una componente que es

perturbada en un mayor grado que las demás y, por lo tanto, es la que interesa monitorizar, pues es de la que mayores conclusiones podremos hallar.

Para eliminar la vibración introducida, se programaron varios algoritmos de suavizado en lenguaje R sobre la componente principal (se utilizó R Studio). Para este ejemplo usaremos los datos de varias repeticiones de flexiones de pecho, centrándonos en la componente y del acelerómetro. Se explicarán los más representativos, mencionando sus pros y contras, así como también cuál es el que finalmente se ha utilizado.

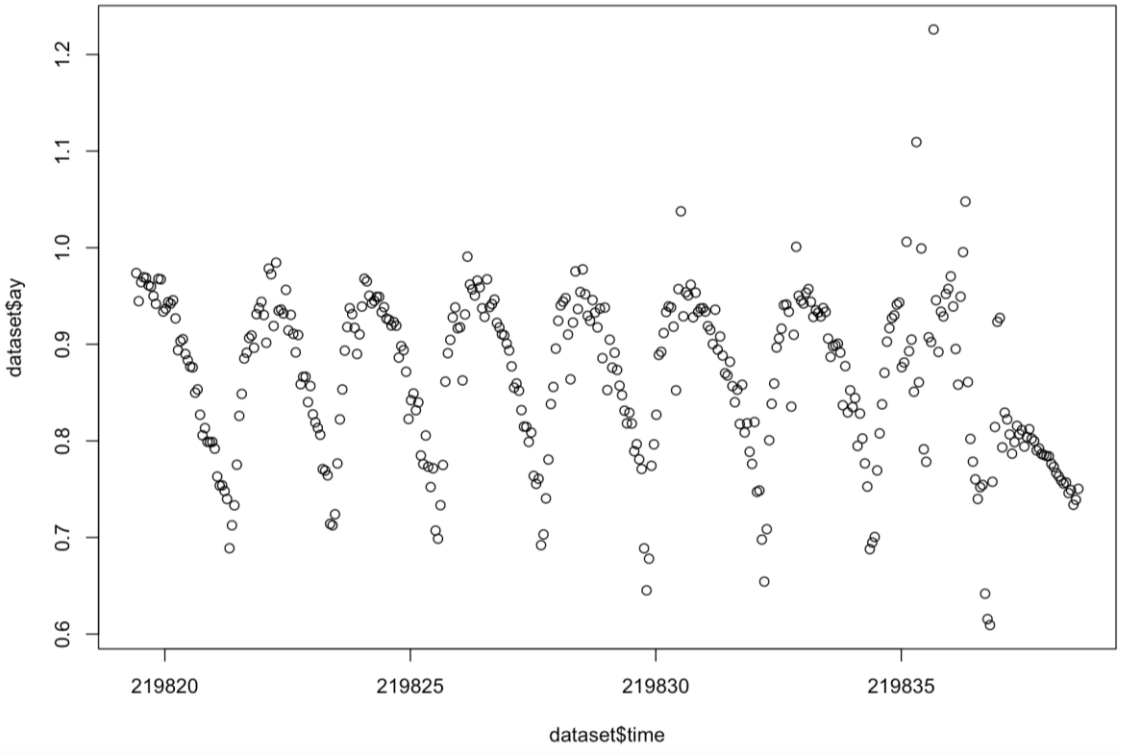


Ilustración 30 - Datos crudo componente y del acelerómetro

6.2 Algoritmos de suavizado

6.2.1 Algoritmo de regresión local (LOESS)

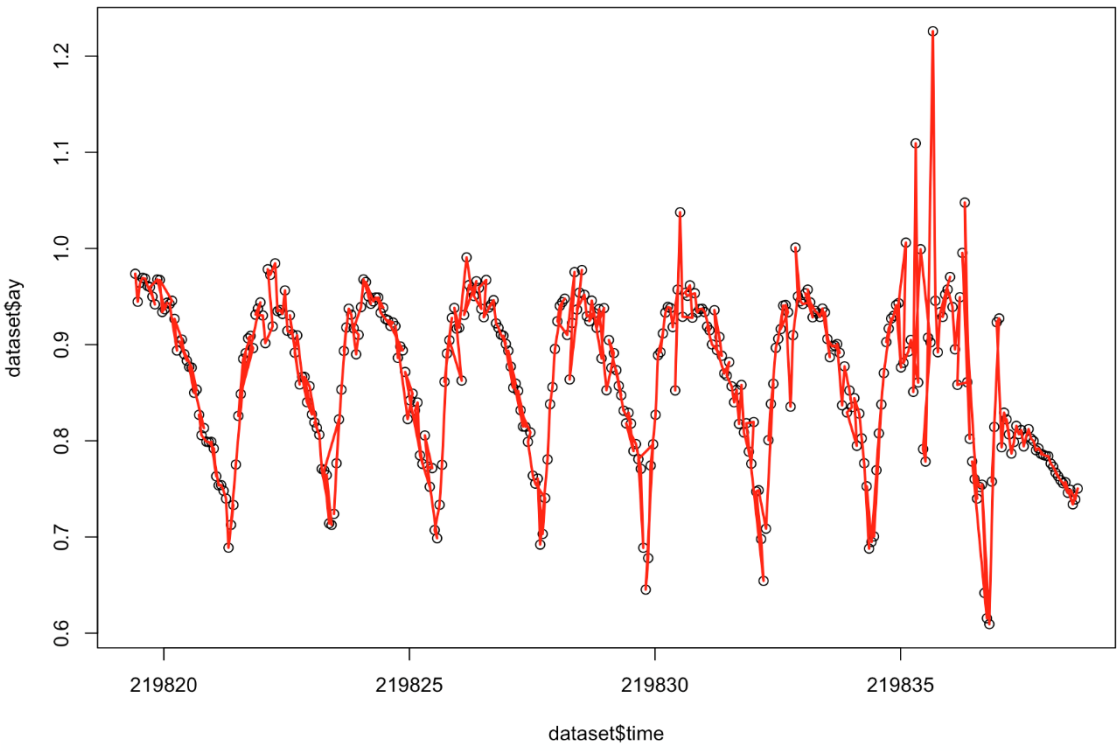


Ilustración 31 - Loess sobre datos en crudo componente y del acelerómetro

El algoritmo de regresión local combina la regresión lineal por mínimos cuadrados con la flexibilidad de la regresión no lineal mediante el ajuste de modelos sencillos sobre subconjuntos locales, creando una función que describe la parte determinista de la variación en los datos punto a punto.

La función tricubo de pesos que es utilizada para asignar el mayor peso a los puntos que se encuentran más cercanos al punto real es la siguiente:

$$w(x) = \begin{cases} (1 - 1|x|^3)^3, & |x| < 1 \\ 0, & |x| \geq 1 \end{cases}$$

El algoritmo de regresión local se utilizó dividiendo la traza de datos en intervalos y, posteriormente, aplicando el algoritmo. La imagen previa muestra la representación del mejor caso que pudimos encontrar, no mejorando en absoluto los datos en crudo provenientes y añadiendo (en algunos casos) ruido extra al modelo.

6.2.2 Suavizado por núcleos (Suavizado de Kernel)

El algoritmo de suavizado por núcleos fija un punto x en el dominio de la función regresora y se define una ventana de suavizado alrededor de ese punto. En caso de que x pertenezca a la recta Real, la ventana será un intervalo que seguirá la fórmula:

$$ventana = (x - h, x + h)$$

El parámetro h tiene un valor fijo llamado ancho de banda. Para realizar la estimación consideramos los puntos x_i dentro de esa ventana:

$$x - h < x_i < x + h$$

Es decir que:

$$|x - x_i| < h$$

Por último, se ha utilizado el estimador por núcleos (Nadayara-Watson), siendo un promedio con pesos dentro de las observaciones y_i dentro de la ventana de suavizado:

$$\hat{\mu}(x) = \frac{\sum_{i=1}^{n} W\left(\frac{x_i - x}{h}\right) y_i}{\sum_{j=i}^{n} W\left(\frac{x_j - x}{h}\right)}$$

W representa la función de pesos. Ésta elige de tal manera que la mayor parte del peso se centre en las observaciones más cercanas a x.

Un posible ajuste es la función bicuadrada. Es la que se ha usado en este ejemplo:

$$W(x) = \begin{cases} (1 - x^2)^2, & -1 \leq x \leq 1 \\ 0, & fuera \end{cases}$$

Asimismo, se podrían elegir también las siguientes variantes:

- Núcleo cuadrático de Epanechnikov.
- Función tricubo.
- Densidad Gaussiana.

Tras la aplicación del método, obtenemos el siguiente resultado:

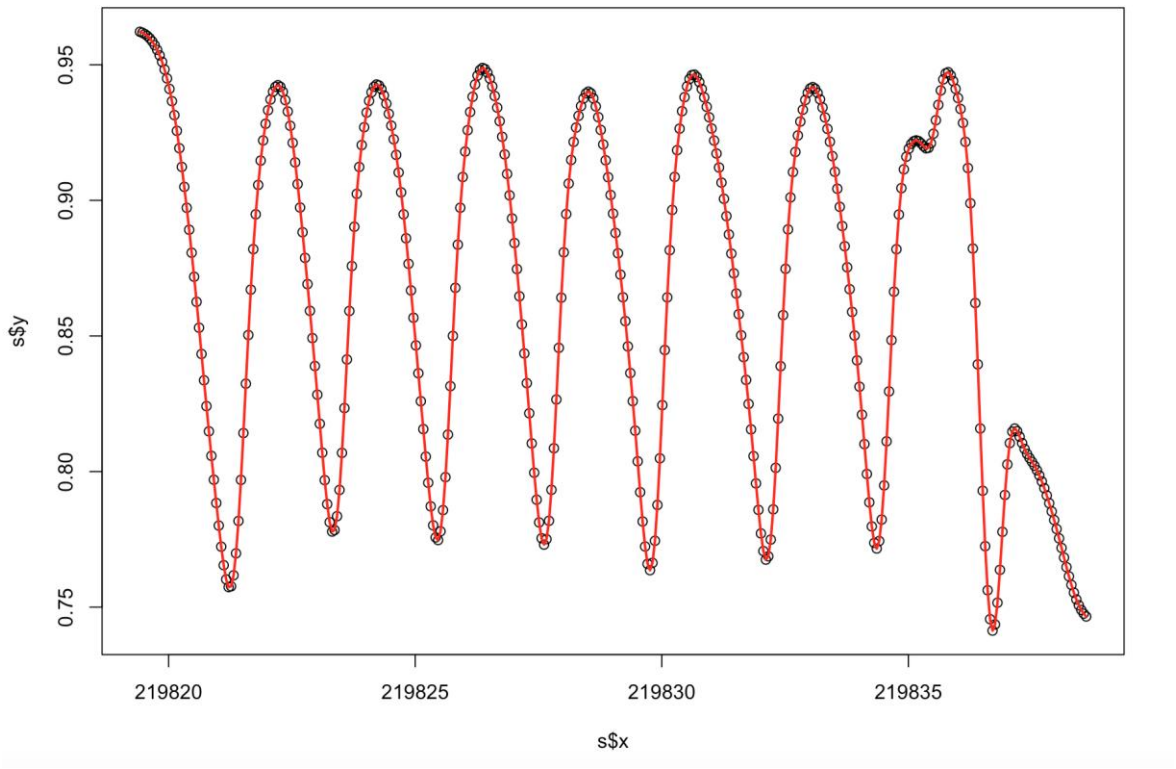


Ilustración 32 - Métodos de Kernel sobre datos en crudo componente y del acelerómetro

Podemos ver claramente como el método suaviza el modelo original, por lo que podría ser un candidato viable.

No obstante, este algoritmo fue desechado por términos de eficiencia, pues, aunque era bastante rápido, la ventana ralentizaba los cálculos innecesariamente.

6.2.3 Spline cúbico

Los splines cúbicos son el tipo de spline más utilizado. La interpolación lineal a trozos es el ejemplo más simple de interpolación, utilizando funciones continuas que, restringidas a cada intervalo de la partición P, son rectas.

Gráficamente, el spline $s(x)$ que interpola a la función f en los puntos $x_o, x_1, \dots, x_n,$ es la poligonal que une los puntos:

$$\big(x_i, f(x_i)\big), \qquad i = 0,1,2,\dots,n$$

Siendo $s_i(x)$ la restricción de $s(x)$ en el intervalo $[x_i, x_{i+1}], i = 0,1,\dots,n-1$, entonces se obtiene que:

$$s_i = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{x+1} - x_i}(x - x_i), \qquad x \in [x_i, x_{i+1}]$$

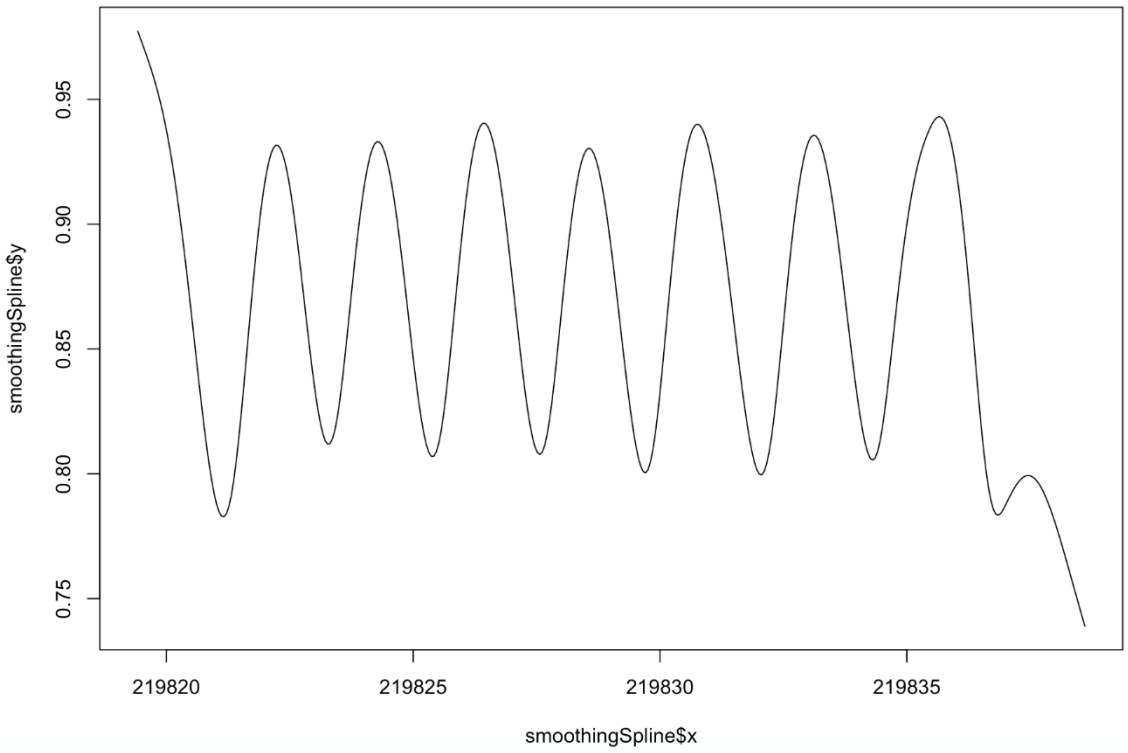


Ilustración 33 - Spline cúbico sobre datos en crudo componente y del acelerómetro

Tal y como se puede apreciar en la ilustración, el spline es capaz de eliminar cualquier ruido en el ejercicio. Asimismo, permite ser implementado de diversas maneras gracias a todas sus variantes, proporcionando unos niveles de eficiencia inmejorables. Es por todo esto por lo que se ha considerado como el óptimo para IASport.

6.3 Análisis de los ejercicios

Para saber cómo aplicar nuestro algoritmo, debemos realizar un análisis previo de varios ejercicios para poder sacar conclusiones de lo que somos capaces de medir, así como también extraer un patrón común a todos ellos.

6.3.1 Identificación de las posiciones de cada ejercicio

Lo primero que debemos determinar son las posiciones comunes a todos los ejercicios.

En un ejercicio, podemos diferenciar entre tres posiciones distintas. Vamos a utilizar el ejemplo flexiones de pecho para una mejor ilustración.

Las posiciones son:

- Posición inicial
- Posición intermedia
- Posición final

La posición inicial es la posición de comienzo del ejercicio, es decir, la posición en la que se encuentra la persona antes de cada repetición.

Ilustración 34 - Posición inicial de un ejercicio

La posición intermedia es entendida como el ecuador de la repetición. En el caso de flexiones de pecho, corresponde al momento en el que los brazos han sido flexionados completamente:

Ilustración 35 - Posición intermedia de un ejercicio

Por último, la posición final corresponde con el final de la repetición, la cual debe coincidir con la posición inicial. En el caso de flexiones de pacho, la posición final se dará cuando vuelva a extender los brazos completamente:

Ilustración 36 - Posición final de un ejercicio

Cabe destacar que una repetición completa es entendida como el proceso de pasar desde la posición inicial a la posición final, pasando por la posición intermedia.

Estas posiciones son comunes a todos los ejercicios, difiriendo en otros parámetros que comentaremos a continuación.

6.3.2 Técnica del ejercicio

En función del ejercicio que se esté realizando, éste tendrá una técnica u otra. La técnica condicionará gran parte de las variables del modelo. Continuando con el caso de flexiones de pecho, su técnica tendría las siguientes restricciones:

- Los brazos deben estar abiertos hasta superar ligeramente la vertical de los hombros.
- Las manos deben estar alineadas al centro de los pectorales.

6.3.3 Tipo de entrenamiento

El tipo de entrenamiento es otra variable que debe ser contemplada. En nuestro ejemplo de flexiones de pecho, podríamos realizarlo según las reglas del tipo de entrenamiento de Hipertrofia, de Quemagrasa, etc. Es decir, un ejercicio puede tener asociados varios tipos de entrenamiento.

Cada una de estas variantes no modificarán la técnica, pero si los tiempos en los que se debe llevar a cabo el ejercicio.

6.3.4 Tiempo de flexión (TF)

El tiempo de flexión (TF) es el tiempo en segundos que se debe tardar en pasar desde la posición inicial a la posición intermedia.

6.3.5 Tiempo de extensión (TE)

El tiempo de extensión (TE) es el tiempo en segundos que se debe tardar en pasar desde la posición intermedia a la posición final.

6.3.6 Tiempo de descanso

El tiempo de descanso entre series o, simplemente, tiempo de descanso, se refiere al tiempo (en segundos) que una persona puede descansar entre serie y serie. Esta variable está íntegramente relacionada con el tipo de entrenamiento. Por ejemplo:

- En Hipertrofia: 240 segundos (4 min).
- En Quemagrasa: 30 segundos (0,5 min).

6.4 Algoritmo IASport

6.4.1 Composición del algoritmo

Para entender el sistema de monitorización de IASport, debemos entender primero el algoritmo que posee.

El algoritmo IASport está compuesto por cuatro procesos asíncronos:

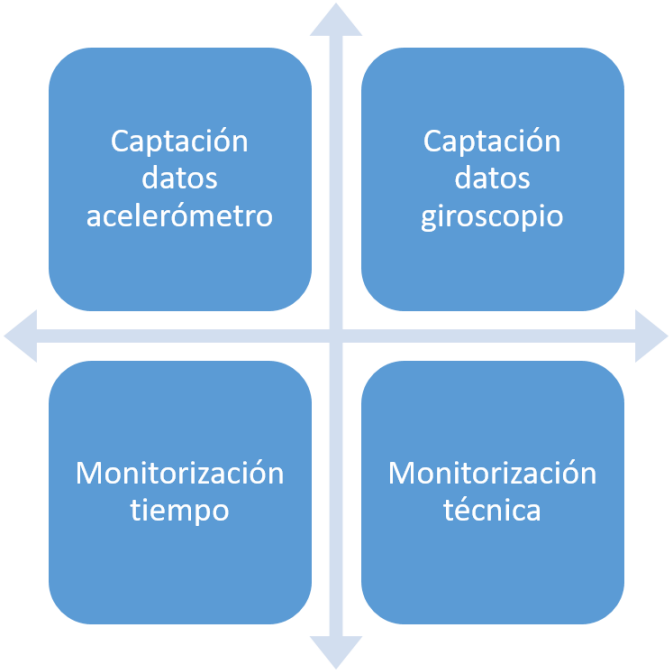


Ilustración 37 - Partes del algoritmo IASport

- La captación de datos del acelerómetro y del giroscopio son dos procesos asíncronos independientes que se ejecutan cada 0,01 segundos, es decir, se realizan 100 lecturas cada segundo (6.000 lecturas cada minuto). Éstos procesos capturan las tres componentes del acelerómetro y giroscopio, así como también la marca temporal que cada lectura posee.
- La monitorización del tiempo y la técnica son dos procesos asíncronos independientes. Éstos controlan que el tiempo que se tarda en la realización del ejercicio y la técnica con la que se lleva a cabo, concuerden con lo establecido por el ejercicio y tipo de entrenamiento, respectivamente. Ambos procesos son ejecutados cada 0.05 segundos, es decir, 20 veces en un segundo (1.200 veces en un minuto).

6.4.2 Parámetros que tiene en cuenta

Tal y como hemos explicado anteriormente, existe un patrón común a todos los ejercicios. No obstante, cada uno debe poseer sus características especiales. Estas particularizaciones se han conseguido gracias a la adición de varios parámetros que el algoritmo IASport tiene en cuenta en tiempo real.

A continuación, enumeramos los parámetros de cada ejercicio:

- limitMinGx: valor mínimo que puede tomar la componente x del giroscopio.
- limitMinGy: valor mínimo que puede tomar la componente y del giroscopio.
- limitMinGz: valor mínimo que puede tomar la componente z del giroscopio.
- errorMinGx: error personalizado que se mostrará cuando en el ejercicio el límite mínimo de la componente x del giroscopio sea rebasada.
- errorMinGy: error personalizado que se mostrará cuando en el ejercicio el límite mínimo de la componente y del giroscopio sea rebasada.
- errorMinGz: error personalizado que se mostrará cuando en el ejercicio el límite mínimo de la componente y del giroscopio sea rebasada.
- limitMaxGx: valor máximo que puede tomar la componente x del giroscopio.
- limitMaxGy: valor máximo que puede tomar la componente y del giroscopio.
- limitMaxGz: valor máximo que puede tomar la componente z del giroscopio.
- errorMaxGx: error personalizado que se mostrará cuando en el ejercicio el límite máximo de la componente x del giroscopio sea rebasada.
- errorMaxGy: error personalizado que se mostrará cuando en el ejercicio el límite máximo de la componente y del giroscopio sea rebasada.
- errorMaxGz: error personalizado que se mostrará cuando en el ejercicio el límite máximo de la componente z del giroscopio sea rebasada.
- restGx: valor de la componente x del giroscopio cuando la persona se encuentra en la posición inicial del ejercicio.
- restGy: valor de la componente y del giroscopio cuando la persona se encuentra en la posición inicial del ejercicio.
- restGz: valor de la componente z del giroscopio cuando la persona se encuentra en la posición inicial del ejercicio.

- limitMinAx: valor mínimo que puede tomar la componente x del acelerómetro.
- limitMinAy: valor mínimo que puede tomar la componente y del acelerómetro.
- limitMinAz: valor mínimo que puede tomar la componente z del acelerómetro.
- limitMaxAx: valor máximo que puede tomar la componente x del acelerómetro.
- limitMaxAy: valor máximo que puede tomar la componente y del acelerómetro.
- limitMaxAz: valor máximo que puede tomar la componente z del acelerómetro.
- restAx: valor de la componente x del acelerómetro cuando la persona se encuentra en la posición inicial del ejercicio.
- restAy: valor de la componente y del acelerómetro cuando la persona se encuentra en la posición inicial del ejercicio.
- restAz: valor de la componente z del acelerómetro cuando la persona se encuentra en la posición inicial del ejercicio.
- component: tal y como se ha mencionado anteriormente, en cada uno de los ejercicios existe una componente principal, entendida como aquella que es más perturbada que el resto cuando se está realizando el ejercicio. Ésta es la que el algoritmo monitorizará ya que es de la que más conclusiones se podrán hallar.
- defaultValue: valor por defecto que se desea que la variable encargada de la calibración posea.
- thresholdTime: margen de tiempo (en segundos). Por ejemplo, si nuestro margen fuera de 0.2 segundos y nuestro tiempo de flexión (TF) fuera 1 segundo, la flexión se consideraría como válida siempre y cuando TF – el tiempo en la posición inicial $\in [0.8, 1.2]$.
- thresholdActivation: la posición inicial, tal y como se ha explicado anteriormente, es entendida como la posición en la que se comienza el ejercicio. El acelerómetro y el giroscopio pueden verse afectados mínimamente por la colocación del dispositivo (puede estar más inclinado, por ejemplo), así como también por la anatomía de la persona. Es por ello por lo que era crucial establecer un margen para evitar este tipo de mala calibración.
- upwardsDisabled: variable que, en caso de estar activa, implicaría que el movimiento se realizará hacia los valores negativos del acelerómetro.
- downwardsDisables: variable que, en caso de estar activa, implicaría que el movimiento se realizará hacia los valores positivos del acelerómetro.

Una vez ya sabemos los parámetros que el algoritmo monitorizará de un ejercicio, necesitamos saber los parámetros que se añaden cuando éste es relacionado con un tipo de entrenamiento.

Recordamos que cada ejercicio debe siempre ir acompañado de un tipo de entrenamiento, y que éste añadirá particularidades.

Parámetros tipo de entrenamiento – ejercicio:

- restBetweenReps: representa el tiempo en segundos del descanso entre repeticiones. La mayoría de veces este valor es 0, no obstante, siempre primamos la flexibilidad, por lo que se podría añadir si llega el caso.
- restBetweenSeries: representa el tiempo en segundos del descanso entre series.
- timePhase1: representa el tiempo en segundos que debe tardar una persona en pasar desde la posición inicial a la posición intermedia, es decir, el tiempo de flexión (TF).
- timePhase2: representa el tiempo en segundos que debe tardar una persona en pasar desde la posición intermedia a la posición final, es decir, el tiempo de extensión (TE).

6.4.3 Detección de errores a partir de los parámetros dados

Una vez tenemos identificados todos los parámetros que somos capaces de medir tanto del ejercicio como de los tipos de entrenamiento a los que está asociado, debemos ser capaces de listar los diferentes errores que el algoritmo IASport será capaz de detectar.

Los errores se pueden clasificar en errores de tiempo o de técnica.

- Errores de tiempo:
    - TF excedido superiormente: error que ocurre cuando el tiempo de flexión es excedido superiormente.
    - TF excedido inferiormente: error que ocurre cuando el tiempo de flexión es excedido inferiormente.
    - TE excedido superiormente: error que ocurre cuando el tiempo de extensión es excedido superiormente
    - TE excedido inferiormente: error que ocurre cuando el tiempo de flexión es extensión inferiormente.
- Errores de técnica:

- limitMinGx excedido: error que ocurre cuando la componente x del giroscopio sobrepasa el parámetro limitMixGx. En tal caso, se mostrará el error almacenado en errorMinGx.
- limitMinGy excedido: error que ocurre cuando la componente y del giroscopio sobrepasa el parámetro limitMixGy. En tal caso, se mostrará el error almacenado en errorMinGy.
- limitMinGz excedido: error que ocurre cuando la componente z del giroscopio sobrepasa el parámetro limitMixGz. En tal caso, se mostrará el error almacenado en errorMinGz.
- limitMaxGx excedido: error que ocurre cuando la componente x del giroscopio sobrepasa el parámetro limitMaxGx. En tal caso, se mostrará el error almacenado en errorMaxGx.
- limitMaxGy excedido: error que ocurre cuando la componente y del giroscopio sobrepasa el parámetro limitMaxGy. En tal caso, se mostrará el error almacenado en errorMaxGy.
- limitMaxGz excedido: error que ocurre cuando la componente z del giroscopio sobrepasa el parámetro limitMaxGz. En tal caso, se mostrará el error almacenado en errorMaxGz.

6.4.4 Funcionamiento

Para explicar el funcionamiento completo, vamos a utilizar el ejemplo del ejercicio flexiones de pecho según el tipo de entrenamiento hipertrofia.

Una vez una persona decide comenzar el ejercicio, los dos procesos asíncronos de captación de datos del acelerómetro y giroscopio almacenan los datos en memoria y son ejecutados cada 0,01 segundos cada uno.

Justo después, los dos procesos asíncronos de monitorización de la técnica y del tiempo se ponen en marcha y se ejecutan cada 0,05 segundos.

El algoritmo de monitorización de la técnica garantiza que nunca se sobrepasen los umbrales del giroscopio. En caso de que alguno sea sobrepasado, se lanzará un error, deteniendo los cuatro

procesos asíncronos. Recodemos que estos umbrales son recogidos de los parámetros de cada ejercicio y tipo de entrenamiento.

El algoritmo de monitorización del tiempo es el más complejo de los tres. En este se encuentra el algoritmo de suavizado, ya que, sin este, las vibraciones añadidas al modelo lo distorsionarían, haciendo que sea imposible su monitorización.

Vamos a captar una flexión en crudo para definir sus fases y explicar cómo lo trata el algoritmo.

Ilustración 38 - Valores en crudo de la componente y del acelerómetro en una repetición de flexión de pecho

El algoritmo de monitorización del tiempo analizaría el fragmento que se ha recogido cada 0,05 segundos. Asimismo, el algoritmo capta los límites de cada una de las repeticiones, así como también la posición intermedia de cada una de ellas.

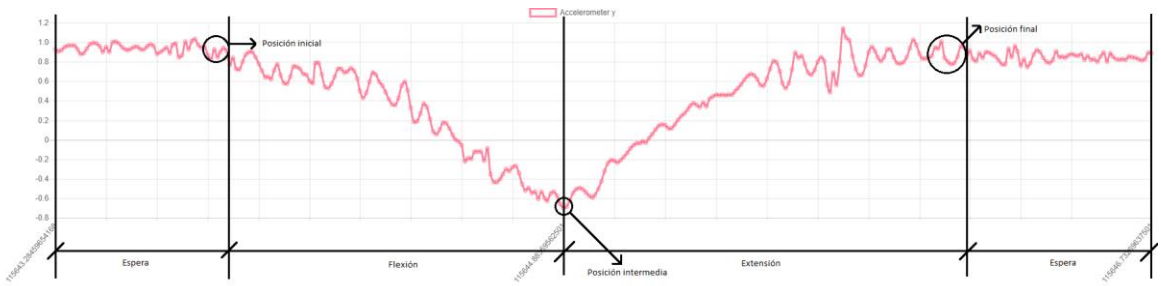


Ilustración 39 - Identificación fases del ejercicio

Cada una de las flexiones tendría el mismo patrón.

Al usar un spline cúbico, debemos definir los intervalos en los que se aplicará. Nosotros hemos elegido cada una de las flexiones como intervalos. El algoritmo es capaz de encontrar los límites de cada una de las flexiones gracias a los parámetros de cada ejercicio y tipo de entrenamiento.

En cada uno de los intervalos, el spline cúbico permite utilizar cualquier otro algoritmo para suavizar dicho intervalo.

En un primer momento, se utilizó el polinomio interpolador de Lagrange. Éste encuentra una función polinómica $L(x)$ de grado k, siendo k el número de conjuntos de puntos -1.

El conjunto de puntos $k+1$ queda definido como:

$$(x_0, y_0), \dots, (x_k, y_k)$$

La función L es la combinación lineal de bases polinómicas de Lagrange definidas como:

$$l_j(x) = \prod_{i=0, i\neq j}^{k} \frac{x - x_i}{x_j - x_i} = \frac{x - x_0}{x_j - x_0} \dots \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x - x_{j+1}}{x_j - x_{j+1}} \dots \frac{x - x_k}{x_j - x_k}$$

Por lo que L se calcula como:

$$L(x) = \sum_{j=0}^{k} y_j l_j(x)$$

La solución con el polinomio de Lagrange nos daba resultados excepcionales. Sin embargo, no era todo lo eficiente que deseábamos que fuera.

Pensamos en numerosas formas de ganar eficiencia en el proceso. Se nos ocurrió que, tiendo los intervalos del spline cúbico y el valor de la posición intermedia, podíamos conseguir cualquier tipo de función utilizando una función cuadrática común:

$$y = ax^2 + bx + c$$

La función cuadrática nos daba resultados igual de exactos que el polinomio de Lagrange, pero de una forma mucho más eficiente.

Una vez la persona ha terminado la flexión actual, el algoritmo detecta que se encuentra en el fin de la repetición. En este momento se realiza el suavizado anteriormente mencionado, así como también una serie de cálculos:

- TF: El tiempo de flexión es calculado mediante la fórmula:

$$TF = tiempo(posición\ intermedia) - tiempo(posición\ inicial)$$

  TF será correcto siempre y cuando $TF \in [timePhase1 - thresholdTime, timePhase1 + thresholdTime]$.

- TE: El tiempo de extensión es calculado mediante la fórmula:

$$TE = tiempo(posición\ final) - tiempo(posición\ intermedia)$$

  TE será considerado como correcto siempre y cuando $TE \in [timePhase2 - thresholdTime, timePhase2 + thresholdTime]$.

Si tanto los cálculos de TF y TE como la técnica (que es monitorizada durante toda la repetición) son correctos, la repetición será dada como correcta.

El sistema de monitorización posee un contador interno del número de repeticiones que el usuario lleva hasta el momento. Cada vez que se realiza una repetición correctamente, el contador es aumentado en una unidad. Éste nuevo valor es comprobado con el número de repeticiones que debía realizar y, si coincide, pondrá fin a la serie. Cabe destacar que si se comete algún error en alguna repetición (sea de técnica o de tiempo), la serie también finalizará, aunque no se haya llegado al número de repeticiones.

Cada vez que la serie finaliza, se envía un resumen de cómo ha ido (número de repeticiones bien hechas, si ha hecho alguna mal y, en caso afirmativo, qué error es el que ha cometido).

Tras enviar el resumen de la serie, sumará una unidad al número de series llevadas a cabo en caso de haberla finalizado. Si este número corresponde con el número total de series que debe realizar, el entrenamiento llegará a su fin. En caso contrario, el algoritmo llamará al método que se encarga de mostrar el cronómetro del tiempo de descanso (en la interfaz aparecerán los segundos hasta que comience la nueva serie). Una vez el cronómetro llegue a cero, se realizará una cuenta atrás para realizar una nueva serie.

7. Diseño

El diseño de un proyecto es crucial para el éxito del mismo. Al comienzo, se barajaron varias hipótesis en cuanto a colores y disposición de los elementos. Tras un largo periodo de investigación y prueba, se detectó que lo mejor hacer uso de colores oscuros con contraste con tonalidades más fuertes para el cliente móvil iOS. Estas tonalidades fuertes se encuentran presentes en la mayoría de productos deportivos, así como también en gimnasios y centros deportivos, ya que son muy vistosos y representan el sacrificio, fuerza y compromiso que el deporte representa. Sin embargo, para la web, la cual sirve como plataforma desde donde los entrenadores y fisioterapeutas administran el sistema, se eligió un diseño más sobrio y limpio jugando con tonalidades más blanquecinas, aunque siempre manteniendo una imagen de marca constante.

7.1 Selección de colores

Tras numerosos tests de usabilidad en la que usuarios heterogéneos participaron, se seleccionó la siguiente paleta de colores:

Ilustración 40 - Paleta de colores iOS

Para los colores de la marca IASport se han utilizado el #21212D y el #C25039. El primer color se ha utilizado para fondos, mientras que el segundo tiene numerosas utilidades tales como resaltar opciones de menú, botones de cancelar, errores en las gráficas, etc.

El color #1C1C26 se ha utilizado para resaltar sutilmente bloques de información del fondo (#21212D).

El color #3093B6 se encuentra a la misma distancia del azul puro que el color #C25039 al rojo puro, lo que hacen a ambos una perfecta combinación. Se utiliza para los botones de confirmación, color de éxito en las gráficas, etc.

Por último, sobre fondos oscuros tales como el actual (#21212D), el color blanco puro puede resaltar demasiado, llegando incluso hasta no ser agradable a la vista. Es por ello por lo que se optó por un color cercano a blanco, pero no tan agresivo (#F5F5F5).

7.2 Logo

El logo de IASport debía reunir todas y cada una de las cualidades que representa. Es por ello que se tenía que resaltar los conceptos deporte, tecnología y vanguardia.

Ilustración 41 - Logo IASport

Tras numerosos bocetos, se optó por la utilización de la herramienta online Logojoy, la cual permite generar y customizar numerosos logos ya predefinidos.

Tras elegir el diseño que mejor se ajustaba a IASport, se eligieron los colores de la marca (#21212D y #C25039).

El círculo que rodea simboliza la comunidad de IASport, nunca cerrada y siempre abierta dando la bienvenida a cualquier persona. La fuente distorsionada con la que están escritas tanto la palabra "IASport" como su lema "High tech training" representa un mundo a caballo entre lo virtual y lo real, pues, aunque el análisis en tiempo real de los datos se realice en el mundo virtual, el deporte que los genera es muy real.

7.3 Fuente

La fuente utilizada para la aplicación móvil, así como también para la web es Raleway. Para los textos se ha hecho uso de su variante Raleway-Light mientras que para los títulos se ha usado en su versión estándar (Raleway).

En la siguiente figura se ilustra una muestra de la misma:

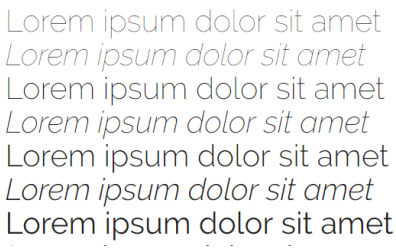


Ilustración 42 - Muestra fuente Raleway

De la fuente podemos dilucidar el carácter vanguardista de IASport, queriendo aportar frescura y modernismo al mundo tan competitivo en el que vivimos.

7.4 Mockups

Previo a la realización de las interfaces tanto de la web como del cliente móvil iOS, se realizaron numerosos mockups de baja fidelidad con la herramienta Balsamiq.

A continuación, se mostrarán algunos mockups significativos. La lista completa se encuentra en el Anexo "Mockups" de este documento.

7.4.1 Cliente móvil iOS

7.4.1.1 Login y registro

Las vistas de login y registro son idénticas, variando únicamente el número de campos que aparecen en cada una de ellas.

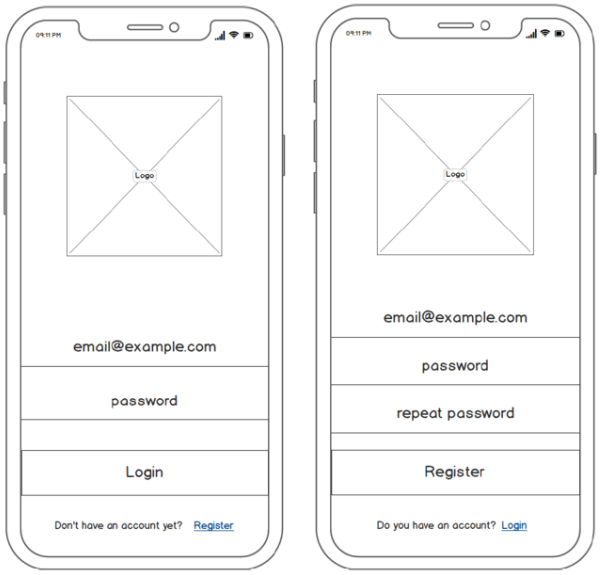


Ilustración 43 - Mockup login y registro

Tras la realización de los mockups, se llevó a cabo la realización de las interfaces. En ellas se aplicaron los colores de marca antes mencionados. Para los botones “Login” y “Registro” se eligió un degradado desde el color de fondo (#21222D) hasta el otro color de marca (#C25039).

Asimismo, se usaron “placeholders” para facilitar al usuario la identificación de cada uno de los campos.

Por último, se añadió una referencia desde la vista de login a la de registro y viceversa.

El login y el registro es una vista sujeta a errores, pues los campos vacíos y/o erróneos están a la orden del día. Es por ello por lo que se diseñó la forma en la que éstos se muestran:

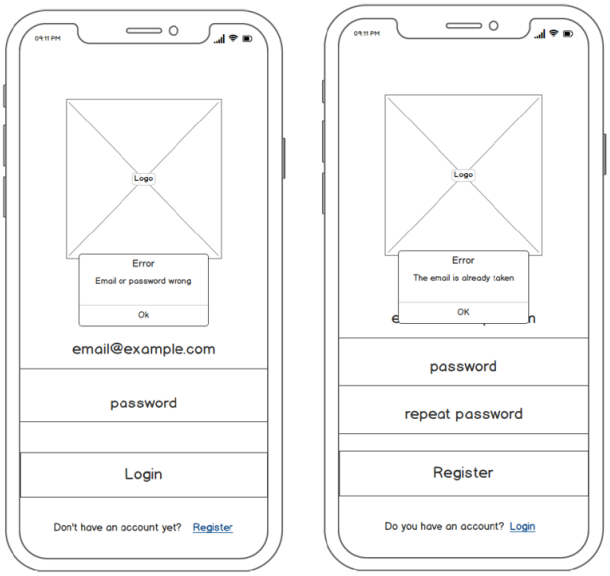


Ilustración 44 – Mockup login y registro con error

7.4.1.2 Menú

Se realizaron numerosos bocetos del menú (con texto e iconos, laterales, superiores…). Finalmente, se llegó a la conclusión de que lo más cómodo era diseñarlo como un menú inferior con sólo iconos, pues los usuarios convencionales están más que familiarizados con éstos al ser la forma más común de manejar la navegación de una aplicación. Asimismo, este tipo de menús es idóneo cuando se tienen pocas opciones como es nuestro caso.

Cabe destacar que el menú debe variar en función de si el rol del usuario es “normal” o “trainer” (que bien puede ser un entrenador o un fisioterapeuta).

En caso de que su rol sea “normal”, el menú sería el siguiente:

Ilustración 45 - Mockup menú usuario “normal”

Las opciones de menú de izquierda a derecha representan las vistas “Perfil”, “Mis rutinas” y “Rutinas” de las que hablaremos más adelante.

Y en caso de ser “trainer”:

Ilustración 46 - Mockup menú usuario "trainer"

La opción añadida en el menú del “trainer” hace referencia a la vista “Custom Training” de la que también se hablará a continuación.

En ambos casos, cuando una opción sea la activa se le asignará el color rojizo principal (#C25039) y aparecerá una línea del mismo color sobre el ítem del mismo color a modo de selector. Las opciones no activan se mantendrían con el color básico para texto (#F5F5F5).

Una vez implementados quedan de la siguiente manera:

Ilustración 47 - Menús iOS

En caso de hacer clic sobre la vista “Rutinas” (p.ej.), el selector y el color cambiarían tal y como se ha explicado anteriormente:

Ilustración 48 - Menú iOS selección ítem rutina

7.4.1.3 Perfil

Una vez un usuario se loga correctamente en el sistema, sería redireccionado a su perfil. En el perfil se ha querido mostrar varias gráficas.

Para la transición de una gráfica a otra se ha usado el elemento PageControl, el cual nos permite realizar scroll lateral a izquierda o derecha según estemos interesados en ir hacia delante o hacia atrás.

Ilustración 49 - Elemento PageControl

7.4.1.4 Detalle rutina a la que se está apuntado

A continuación, se muestra el diseño de "Detalle rutina" cuando el usuario está apuntado a la misma:

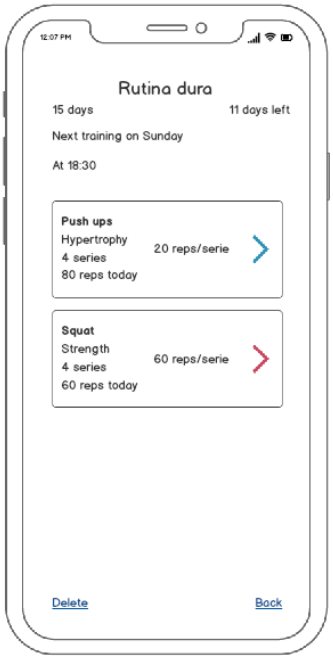


Ilustración 50 - Mockup detalle rutina a la que se está apuntado

En ella se muestra información sobre la rutina y, posteriormente, se muestra una tabla con todos los ejercicios.

Cada uno de los ejercicios tendrá el siguiente diseño:

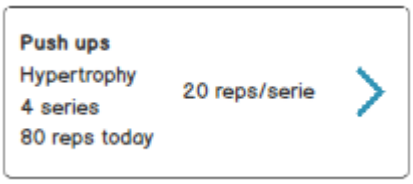


Ilustración 51 - Mockup ejercicio terminado rutina apuntada

En él aparece una flecha de color azulón de la marca (#3093B6), lo que se usa para indicar que hoy ya se ha llegado al número de repeticiones que se debían realizar, ya que se han realizado ochenta repeticiones de un total de ochenta (4 series por 20 repeticiones/serie = 80 repeticiones).

En caso de que hoy no se hayan realizado todas las repeticiones, la flecha aparecerá del color rojizo de marca (#C25039):

Ilustración 52 - Mockup ejercicio no terminado rutina apuntada

7.4.1.5 Train

Train es la vista donde se realizan todos y cada uno de los ejercicios de una rutina.

Se diseñó de la siguiente manera:

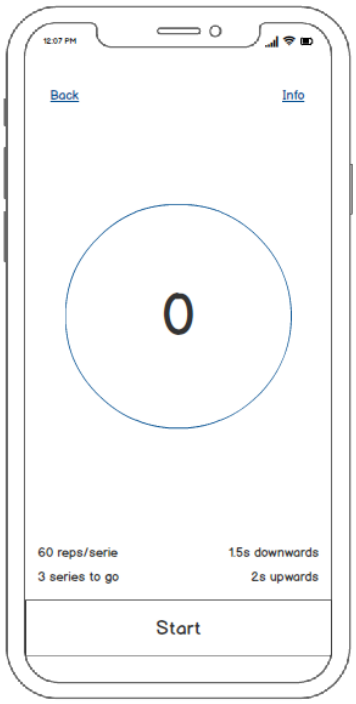


Ilustración 53 - Mockup train

Se pensó que el círculo que rodea al número central (número de repeticiones realizadas) se cargara realizando una transición.

8. Funcionalidades

En este capítulo se hablarán de todas las funcionalidades implementadas. Se hará una distinción entre las funcionalidades de la aplicación móvil y la web, así como también aquellas que sean comunes a ambas.

Recordamos que en IASport existen tres roles de usuario; usuario no identificado, usuario normal y trainer. Cada uno de ellos puede acceder a una serie de métodos y realizar una serie de acciones propias de su rol. Un trainer puede realizar las acciones de un usuario normal y un usuario normal las de un usuario no identificado. Para una mayor claridad, en el título de cada funcionalidad se añadirá entre paréntesis el rol más restrictivo que se precisa.

8.1 Funcionalidades comunes

8.1.1 Registro y autenticación de usuarios (usuario no identificado)

Esta funcionalidad es crucial para identificar a los usuarios, así como también permitirles a los nuevos usuarios apuntarse a la plataforma. Está íntegramente relacionada con los aspectos de seguridad que la aplicación posee. Para ver el detalle completo véase el apartado de Seguridad. Una vez un usuario es logado, podrá realizar las acciones que su rol le permita.

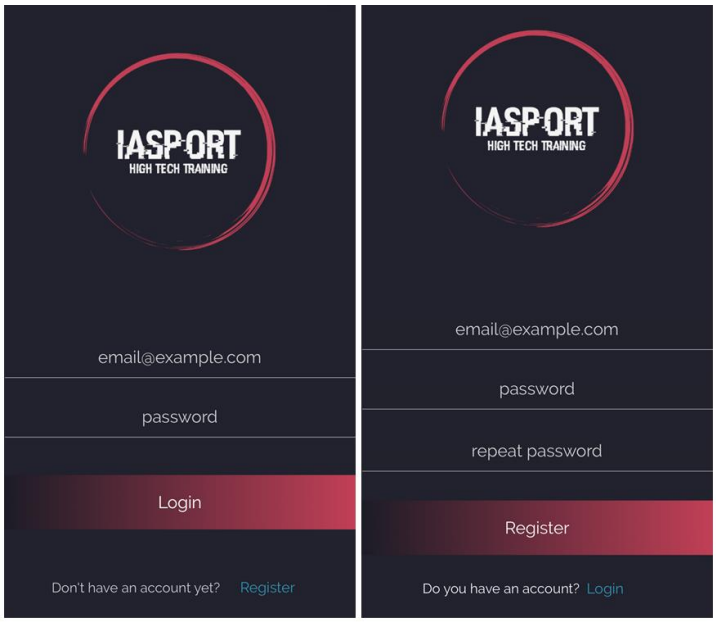


Ilustración 54 - Login y registro en iOS

Puesto que esta vista está sujeta a cuantiosos errores producidos por una mala introducción de credenciales, era preciso tenerlo en cuenta para el diseño:

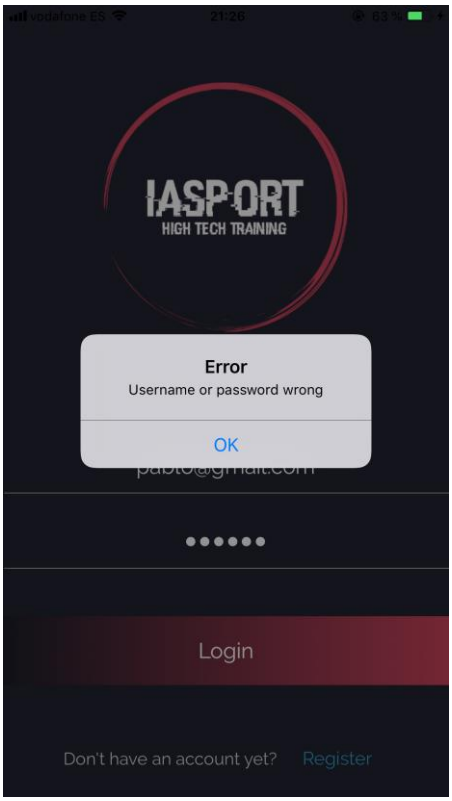


Ilustración 55 – Login con error en iOS

Análogamente, en la parte web se reflejan de la siguiente manera:

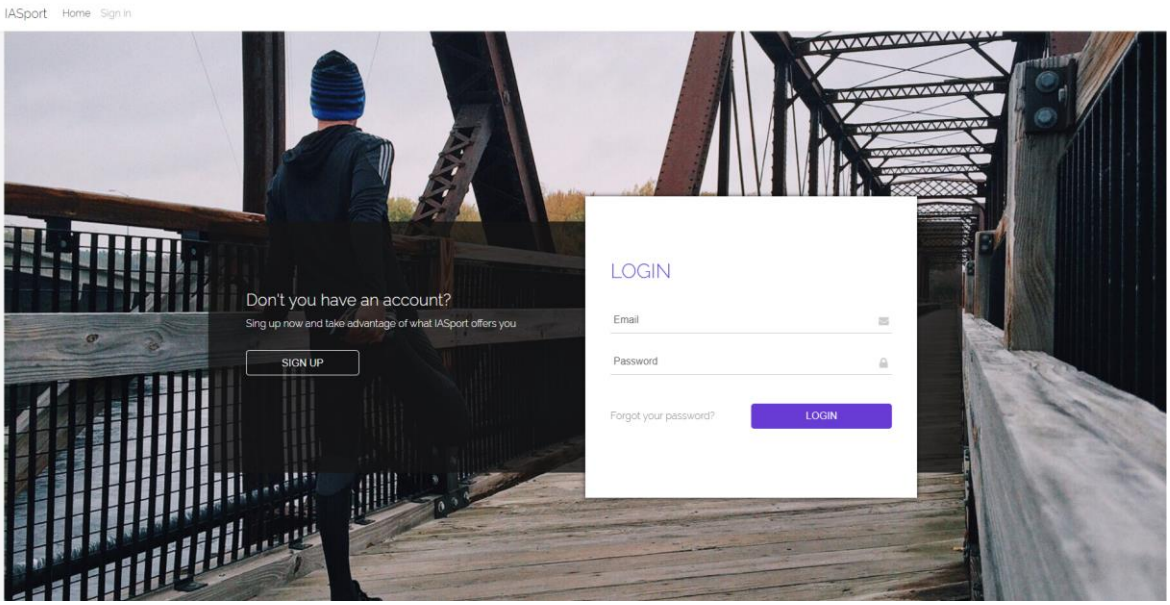


Ilustración 56 – Login en web

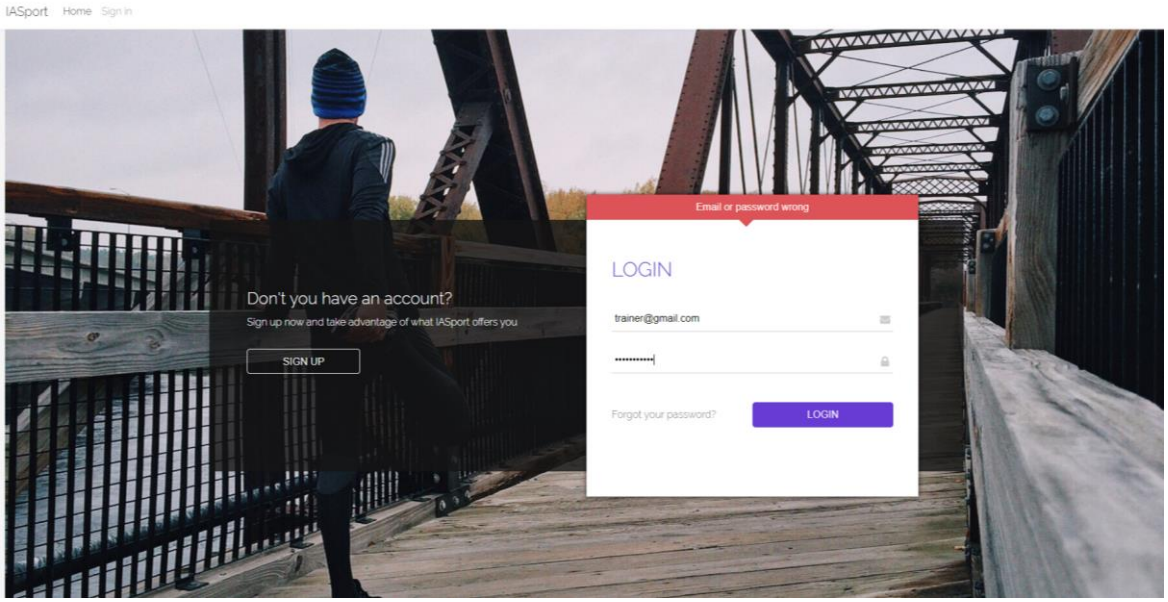


Ilustración 57 - Login en web con error

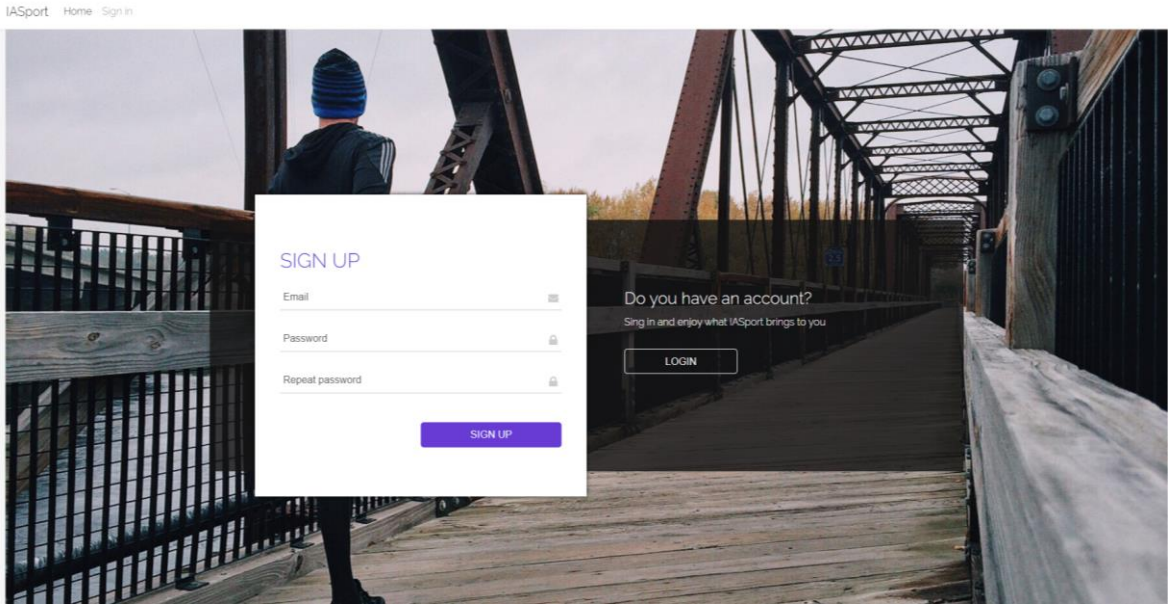


Ilustración 58 – Registro en web

8.1.2 Visualizar rutinas (usuario normal)

Un usuario podrá ver todas y cada una de las rutinas, tanto a las que está apuntado como a las que no.

Previamente se diseñó la forma en la que mostrar una sola rutina. En ella debía aparecer el nombre de la rutina, así como también el número de sesiones que se deben realizar de la misma para que sea considerada como finalizada. Se pensó de la siguiente manera:

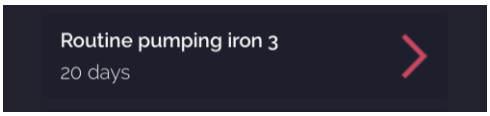


Ilustración 59 – Diseño rutina en iOS

También se añadió el icono de la flecha en la parte derecha, pues cuando se pulse sobre cualquiera de ellas se mostrará la vista “Detalle rutina” con todos sus datos. De esta vista hablaremos a continuación.

Por último, se crearon las vistas de “Mis rutinas” y “Rutinas” añadiendo una tabla que contuviera todas las rutinas:

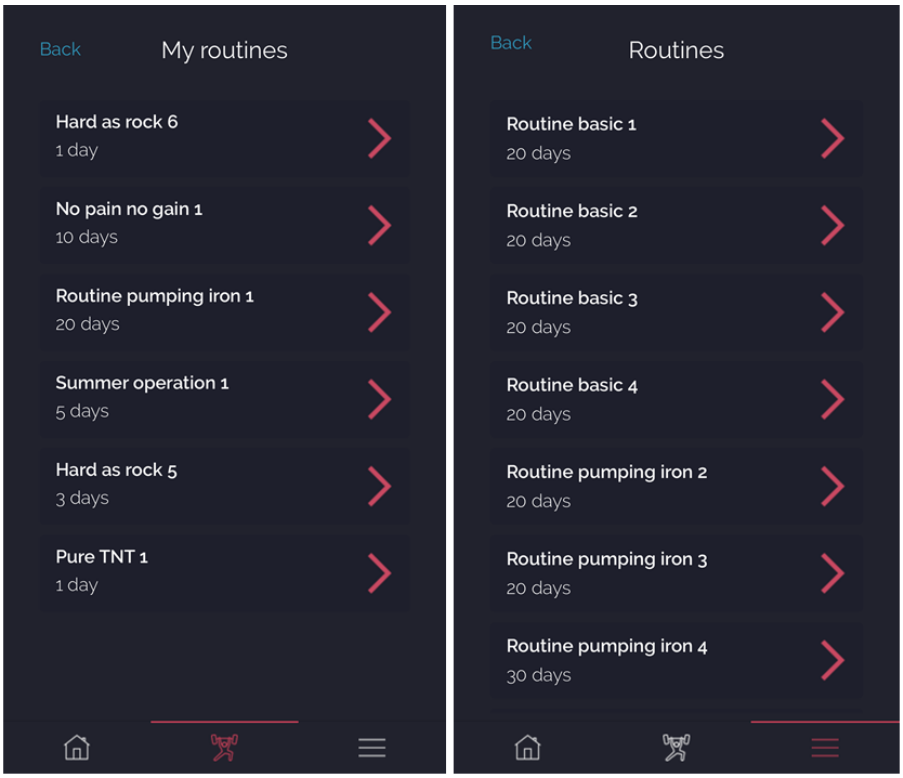


Ilustración 60 - Mis rutinas y rutinas en iOS

Su representación en web sería la siguiente:

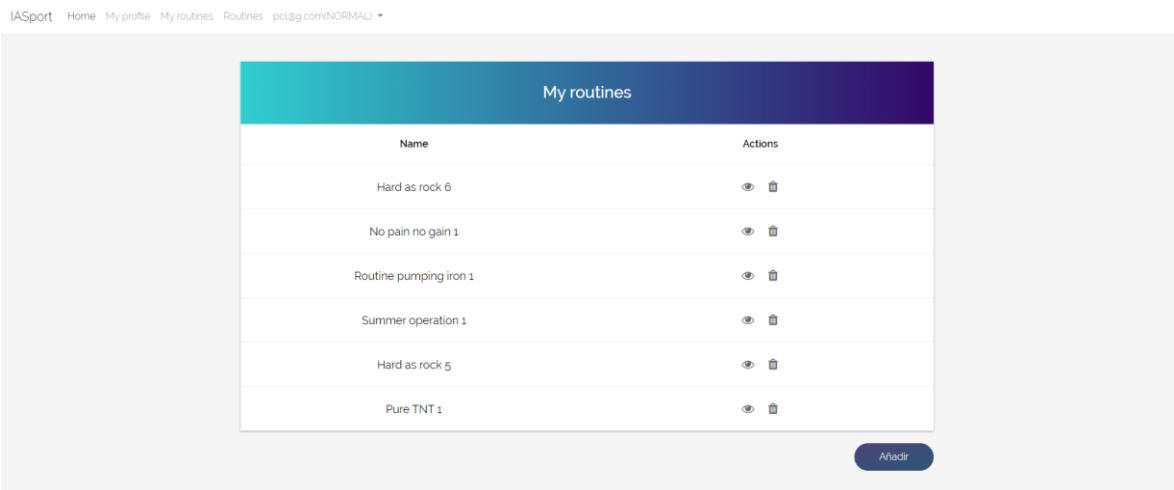


Ilustración 61 – Mis rutinas en web

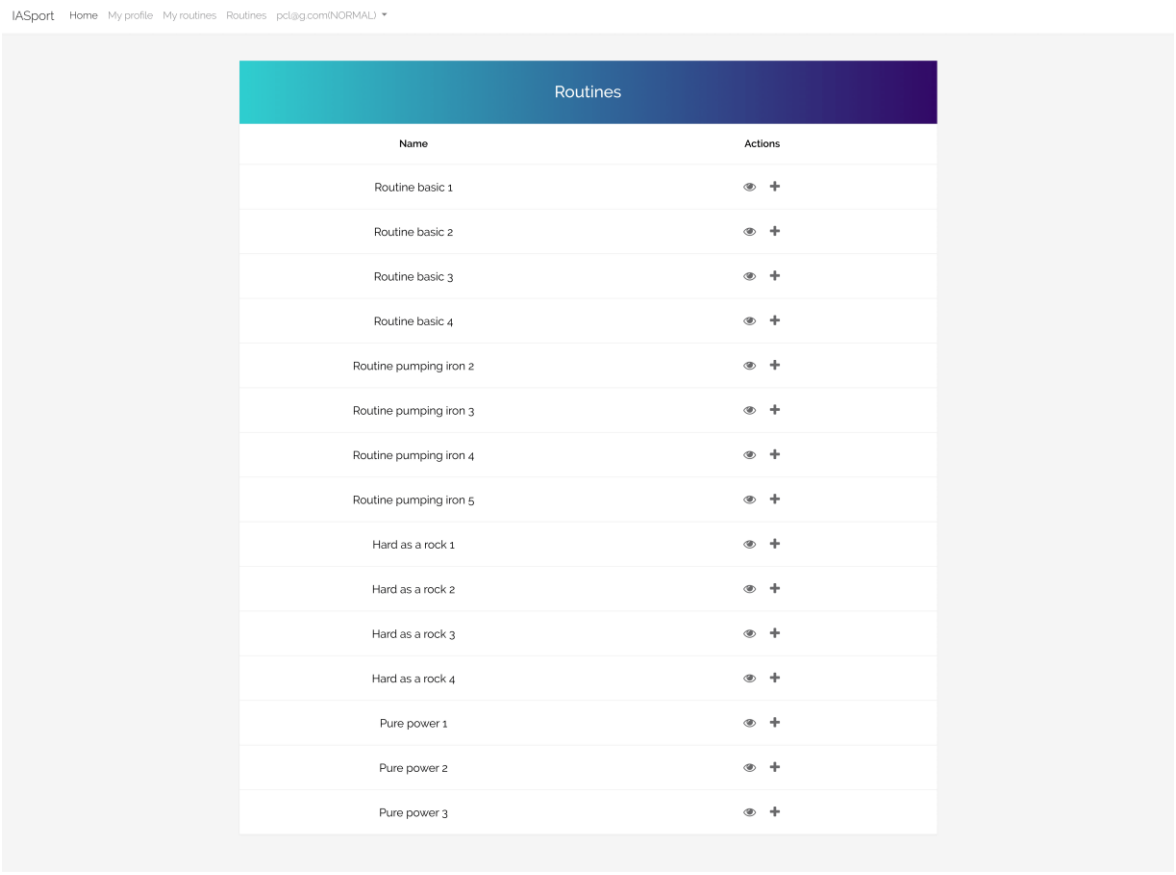


Ilustración 62 - Rutinas en web

8.1.3 Visualizar detalle rutina a la que se está apuntado (usuario normal)

Una vez un usuario pulse sobre una rutina a la que está apuntado, visualizará todos y cada uno de los detalles de la misma.

A continuación, se muestra el diseño de “Detalle rutina”:

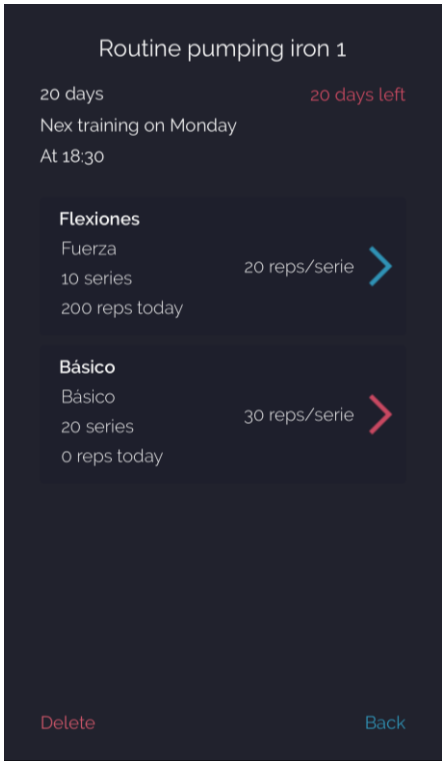


Ilustración 63 – Detalle rutina a la que se está apuntado en iOS

En ella se muestra el nombre de la rutina como título (“Routine pumping iron 1”), el número de sesiones que debe ser realizada (“20 days”), el número de sesiones que le quedan por realizar (“20 days left”), lo que implica no se ha realizado ningún día de la rutina aún, así como también el siguiente día que le toca entrenar (“Next training on Monday”) junto con la hora (“At 18:30”).

Posteriormente, se muestra una tabla con todos los ejercicios.

Cada uno de los ejercicios tendrá el siguiente diseño:

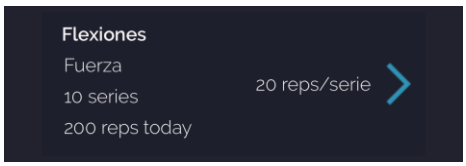


Ilustración 64 – Ejercicio terminado en iOS

En él se muestra el nombre del ejercicio (“Flexiones”), el tipo de entrenamiento en el que se va a realizar (“Fuerza”), el número de series que se debe realizar (“10 series”), el número de repeticiones

que se han llevado a cabo hoy (“200 reps today”) y el número de repeticiones de cada serie (“20 reps/serie”). Por último, se muestra una flecha para indicar que si pulsamos sobre el ejercicio nos llevará a la vista donde se llevará a cabo, llamada “Train” de la que se hablará y de la que hablaremos a continuación. La flecha aparecerá con el color rojo de la marca si todavía no se han realizado el número de repeticiones correspondientes en el día actual.

El detalle de una rutina en la web se mostraría de la siguiente manera:

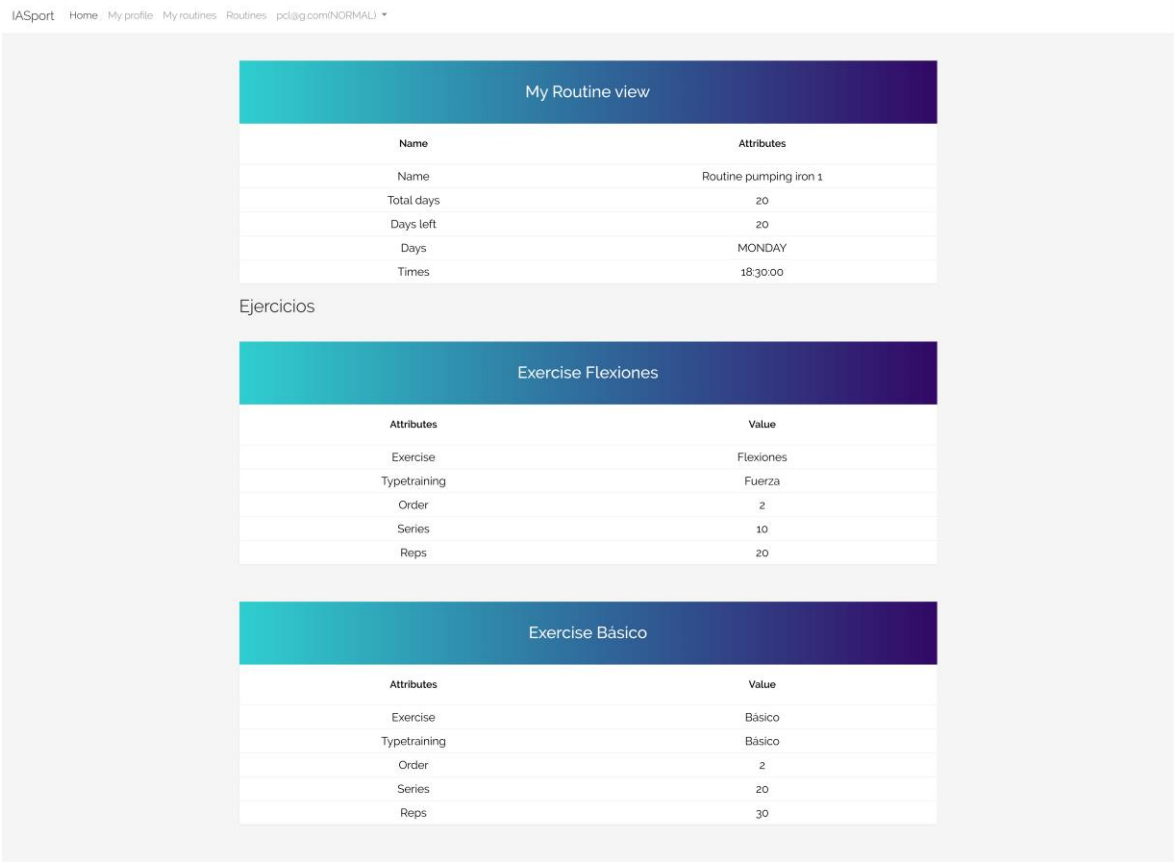


Ilustración 65 – Detalle rutina a la que se está apuntado en web

8.1.4 Visualizar detalle rutina a la que no se está apuntado (usuario normal)

Un usuario debe poder conocer todos los detalles de una rutina a la que no está apuntado todavía. Una vez pulse en la rutina de la que quiere conocer los detalles, se le mostrará la siguiente vista en iOS:

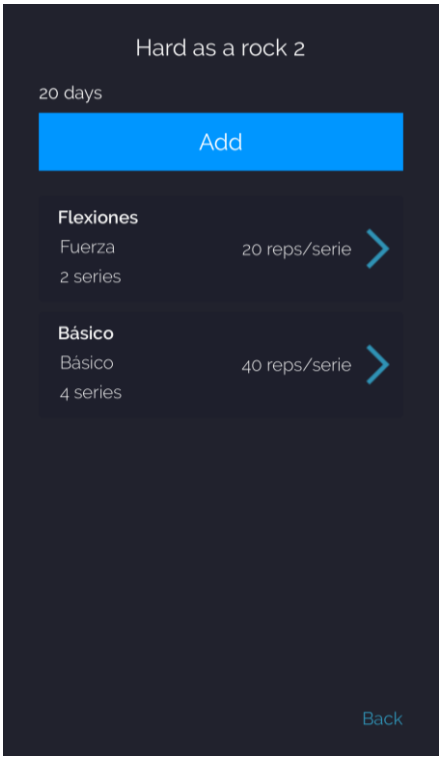


Ilustración 66 – Detalle rutina a la que no se está apuntado en iOS

En la vista se puede apreciar la información que una rutina posee.

En la parte superior izquierda se encuentra el número de sesiones (días) que debe ser realizada para que se considere como finalizada ("20 days"), más abajo, un botón ("Add") que sirve para apuntarse a la misma (del que hablaremos a continuación), así como también la lista de ejercicios. Cada uno de los ejercicios se encuentra enmarcado. Esta rutina en concreto tendría dos ejercicios.

La forma en la que se muestran los ejercicios en esta vista ha cambiado ligeramente, ya que se han eliminado campos que sólo disponemos si estamos apuntados a la rutina:

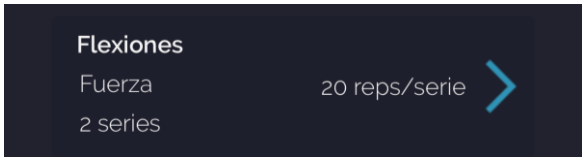


Ilustración 67 – Ejemplo ejercicio rutina no apuntada en iOS

En él se muestra el nombre del ejercicio (“Flexiones”), el tipo de entrenamiento en el que se realizará (“Fuerza”), el número de series que se realizará (“2 series”), así como también el número de repeticiones de cada serie (“20 reps/serie”).

Además, aparece la misma flecha que en los ejercicios de las rutinas a las que estoy apuntado. En caso de que el usuario pulse sobre cualquier de ellos, se mostrará un error diciendo que debe estar apuntado a la rutina para poder entrenar:

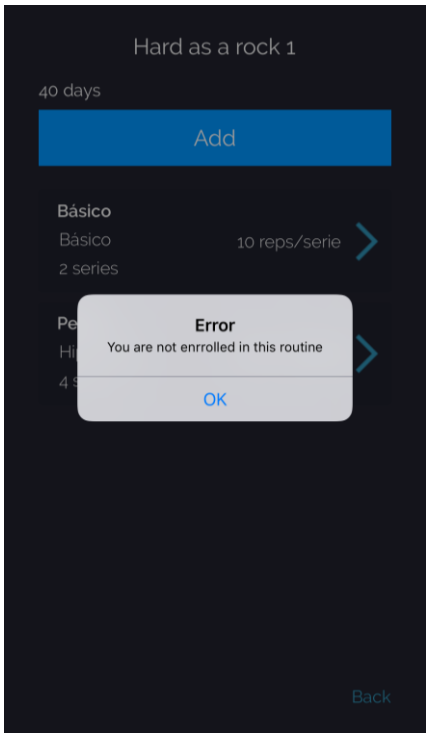


Ilustración 68 – Detalle rutina no apuntada con error en iOS

Análogamente, toda esta información es mostrada en la web:

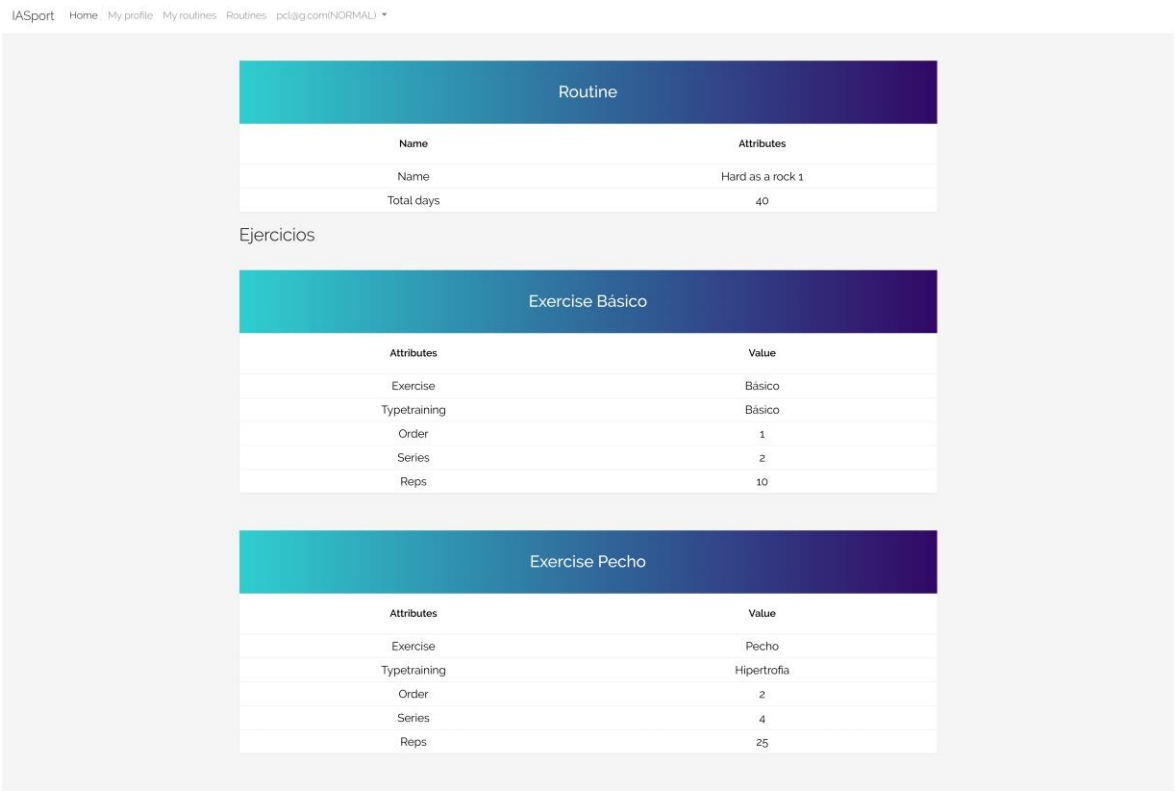


Ilustración 69 – Detalle rutina a la que no se está apuntado en web

8.1.5 Apuntarse a una rutina (usuario normal)

Un usuario al que le interese una rutina a la que no está apuntado, puede apuntarse indicando los días y a las horas que desea realizarla.

Tanto en la web como en el cliente iOS aparece un botón “Add” el cuál le solicitará la información antes mencionada.

Estudiemos el caso concreto del cliente iOS:

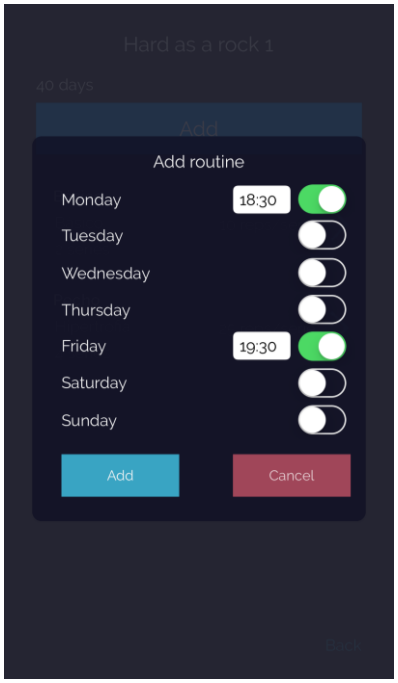


Ilustración 70 – Apuntarse a una rutina en iOS

Una vez hemos pulsado sobre el botón “Add” de la vista de detalle de la rutina a la que no estábamos apuntados, aparecerá un pop up con todos los días de la semana para que se seleccionen aquellos en los que se vaya a entrenar. Una vez seleccionados, aparecerá un cuadro de texto en el que se debe escribir la hora a la que se desea hacerlo.

Una vez toda la información haya sido cumplimentada, el proceso de apuntarse a una rutina se completará al pulsar sobre el botón “Add” del pop up. Si todo ha ido bien, el usuario será redirigido a la lista de las rutinas a las que está apuntado donde aparecerá también a la que se acaba de apuntar.

8.1.6 Desapuntarse de una rutina (usuario normal)

Un usuario que desea desapuntarse de una rutina tiene la posibilidad de hacerlo.

En caso del cliente iOS al final de la vista “Detalle rutina” aparecen dos botones “Delete” y “Back” para desapuntarse de la rutina y para volver a la lista de “Mis rutinas” respectivamente:

Ilustración 71 – Botón delete para desapuntarse de una rutina en iOS

Al pulsar sobre “Delete” el usuario se desapuntará automáticamente de la rutina y será redireccionado a “Mis rutinas”. Cabe destacar que la rutina de la que acaba de desapuntarse se encontrará disponible en “Rutinas”.

Por el contrario, para eliminar una rutina desde la web, se deberá hacer desde la lista de “Mis rutinas” y darle al icono de la papelera.

8.1.7 Perfil (usuario normal)

Una vez un usuario se loga correctamente en el sistema, sería redireccionado a su perfil. En el perfil se ha querido mostrar un conjunto de gráficas principales que representan la progresión del usuario. Las gráficas varían de la web al cliente iOS.

Gráficas cliente iOS:

- Repeticiones que ha realizado por tipo de entrenamiento.
- Repeticiones realizadas correcta e incorrectamente por fecha (días).
- Número total de repeticiones realizadas correcta e incorrectamente.
- Número de errores por su tipo.

A continuación, se muestran las vistas realizadas para albergar las cuatro gráficas, simulando el scroll lateral gracias al elemento PageControl mencionado en el apartado de Diseño:

Ilustración 72 – Gráficas perfil página 1 y 2 en iOS

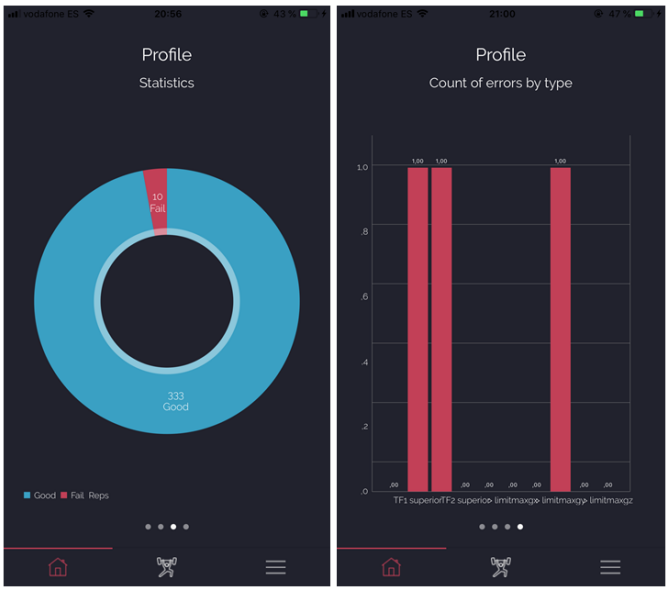


Ilustración 73 - Gráficas perfil página 3 y 4 en iOS

Análogamente, en la web se pueden encontrar las siguientes gráficas:

- Número de repeticiones correctas por tipo de entrenamiento
- Número total de repeticiones correctas
- Número total de repeticiones incorrectas

- Repeticiones correctas vs incorrectas por tipo de entrenamiento
- Top 3 tipos de entrenamiento
- Número de repeticiones por fecha (días)
- Top 3 ejercicios y su tipo de entrenamiento
- Número de errores cometidos por su tipo

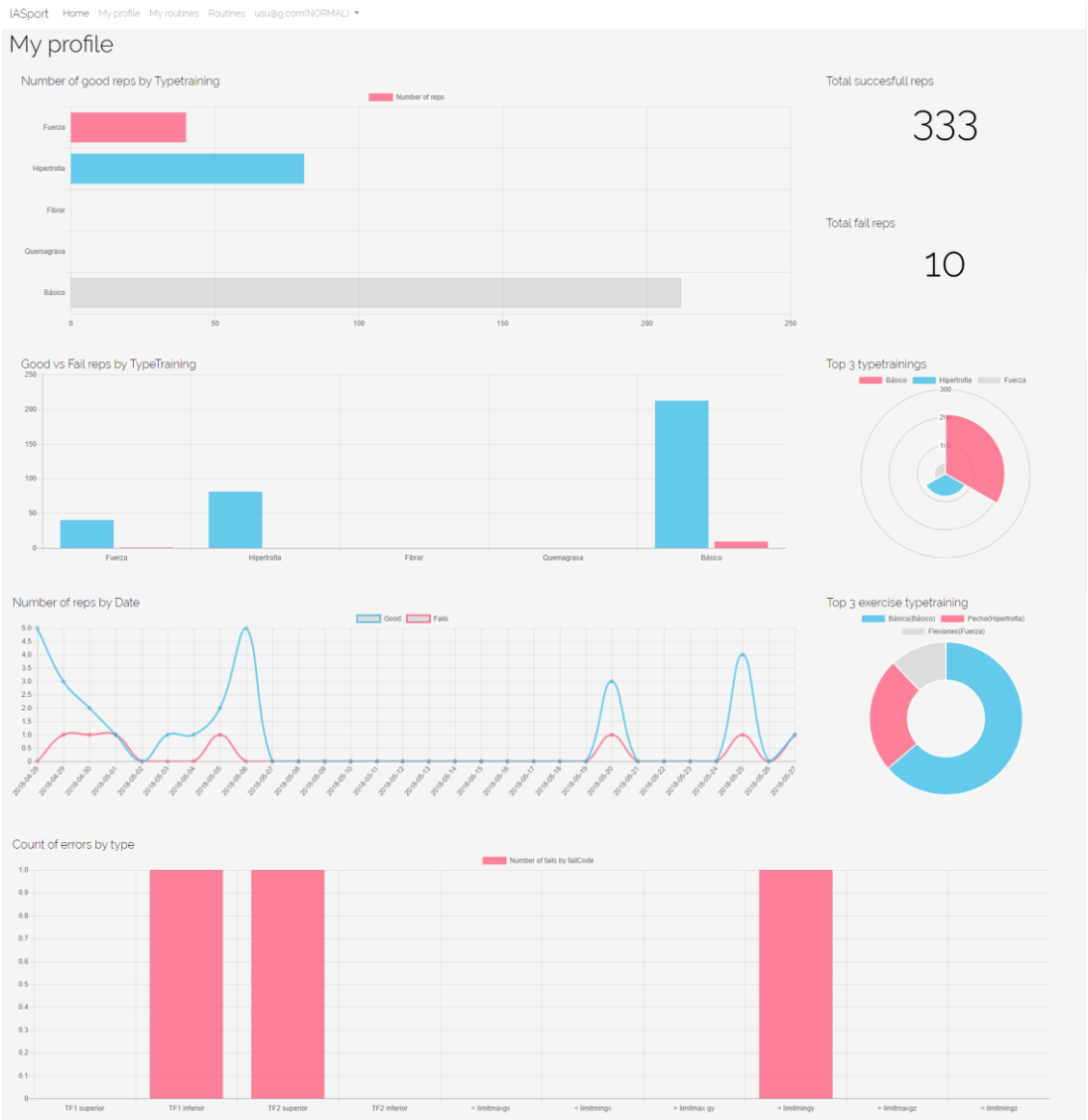


Ilustración 74 – Perfil en web

8.2 Funcionalidades web

8.2.1 Custom trainings

Una de las cualidades principales de IASport es la flexibilidad que ofrece, pudiendo añadir cualquier ejercicio. Existe una cantidad ingente de parámetros para crear un ejercicio. Es por ello por lo que se precisa de un lugar donde poder crear ejercicios personalizados y visualizar sus datos para poder rellenar todos y cada uno de los parámetros de forma precisa.

Recordamos que esta funcionalidad está restringida a los trainers.

8.2.1.1 Crear custom trainings (trainer)

Cada trainer puede disponer de una lista de custom trainings diferenciados por un nombre. Para crear uno, basta con asignarle un nombre desde la web:

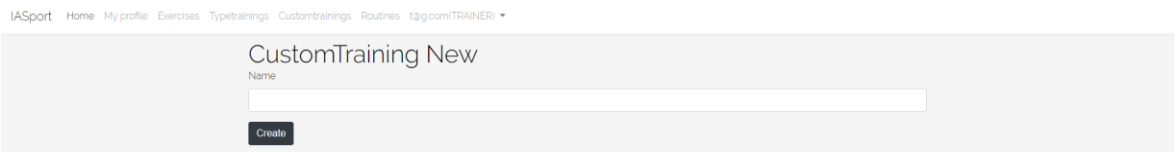


Ilustración 75 - Crear custom training en web

Cabe destacar que los custom trainings son privados para cada trainer, por lo que cada uno tendrá los suyos propios no pudiendo acceder a los de los demás.

8.2.1.2 Visualizar lista custom trainings (trainer)

El trainer debe poder acceder a la lista de custom trainings que ha creado hasta el momento. Para ello, se ha provisto a la web de una sección llamada Custom training:

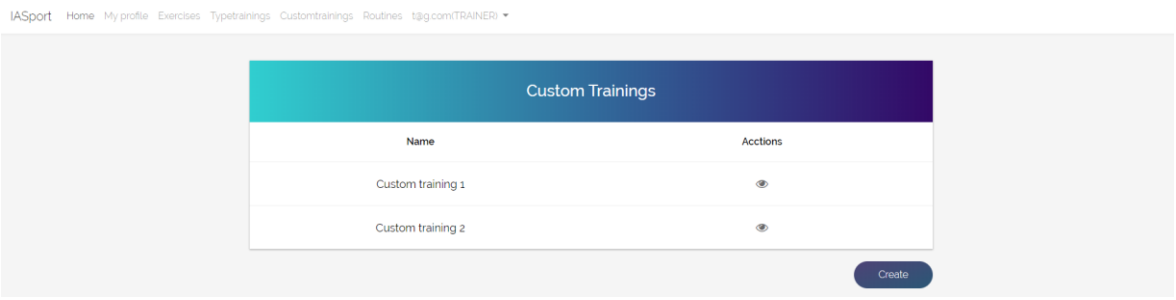


Ilustración 76 - Lista custom trainings en web

8.2.1.3 Visualizar datos custom trainings (trainer)

Una vez el trainer ha entrenado en un custom training específico desde el cliente móvil iOS (comentaremos esta funcionalidad más adelante), todos los datos del acelerómetro y giroscopio habrán sido capturados y enviados a la API. Una vez la sincronización de todos los datos haya sido completada, desde la web se podrán visualizar gracias a 6 gráficas para el acelerómetro y giroscopio (una para cada componente x, y, z) tal y como se muestra a continuación:

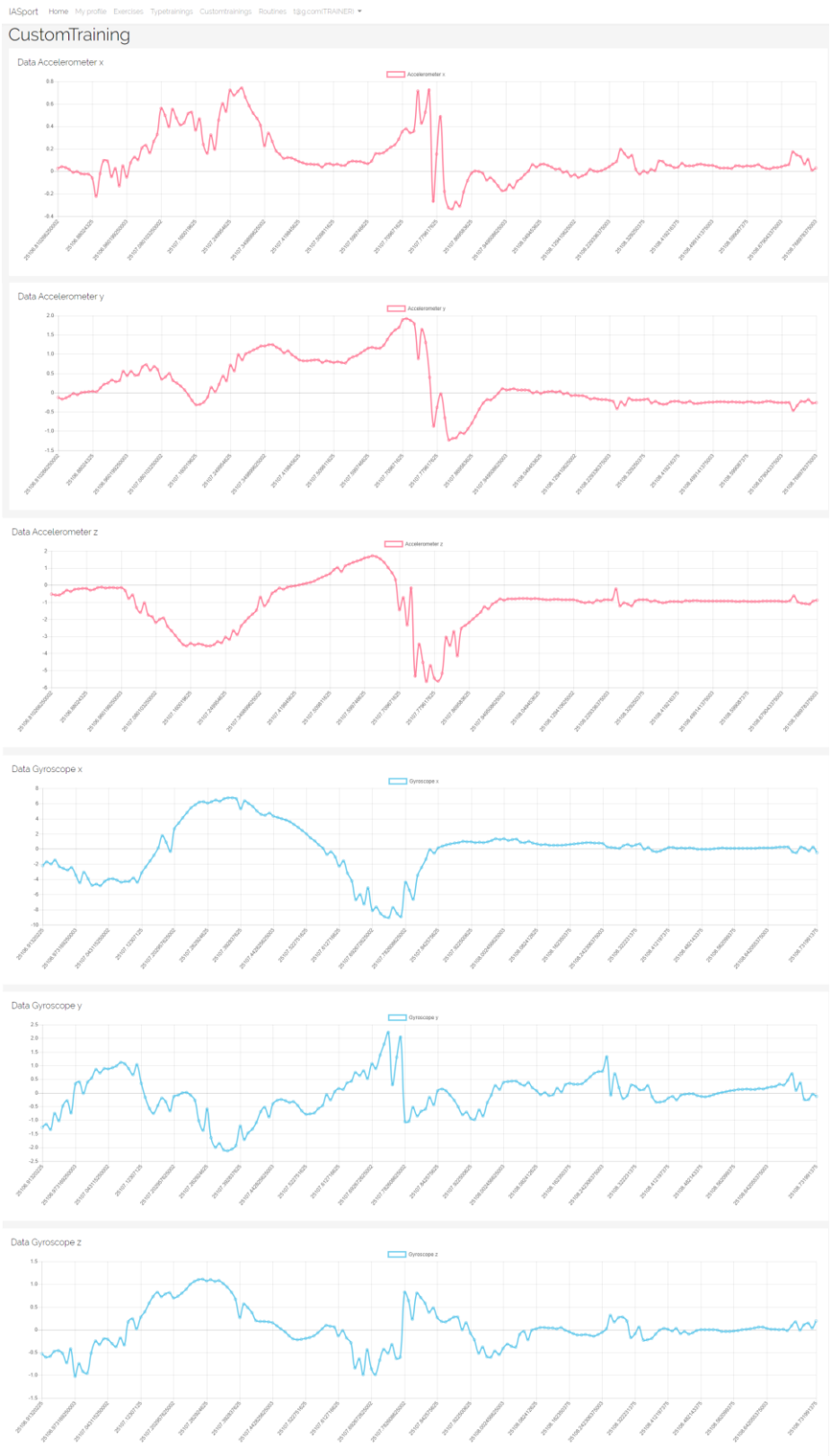


Ilustración 77 - Visualización datos custom training en web

8.2.2 Ejercicios

8.2.2.1 Crear ejercicio (trainer)

Crear un ejercicio es una de las funcionalidades más básicas que el sistema posee. Existe un formulario sólo accesible por trainers que, tras ser rellenarlo, un nuevo ejercicio se añade a la plataforma:

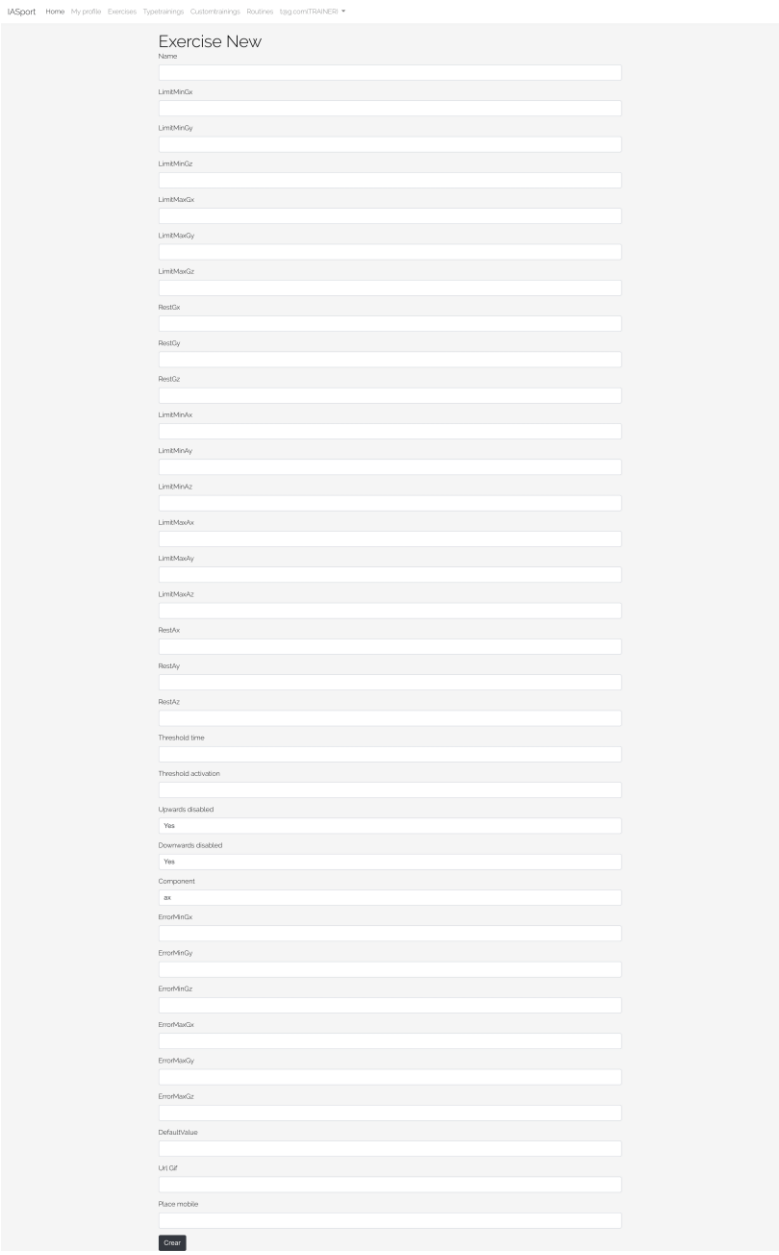


Ilustración 78 - Formulario crear ejercicio en web

Cabe destacar que el formulario posee los 33 parámetros que un ejercicio necesita.

8.2.2.2 Editar ejercicio (trainer)

Una vez un ejercicio es añadido, no puede ser un objeto inamovible, pues este debe poder ser editado para su calibración o actualización. En el cliente web existe un formulario el cual posee los 33 parámetros con los valores actuales del ejercicio.

Este formulario se puede editar como el trainer crea conveniente. Tras su edición, el ejercicio se habrá actualizado con los nuevos valores.

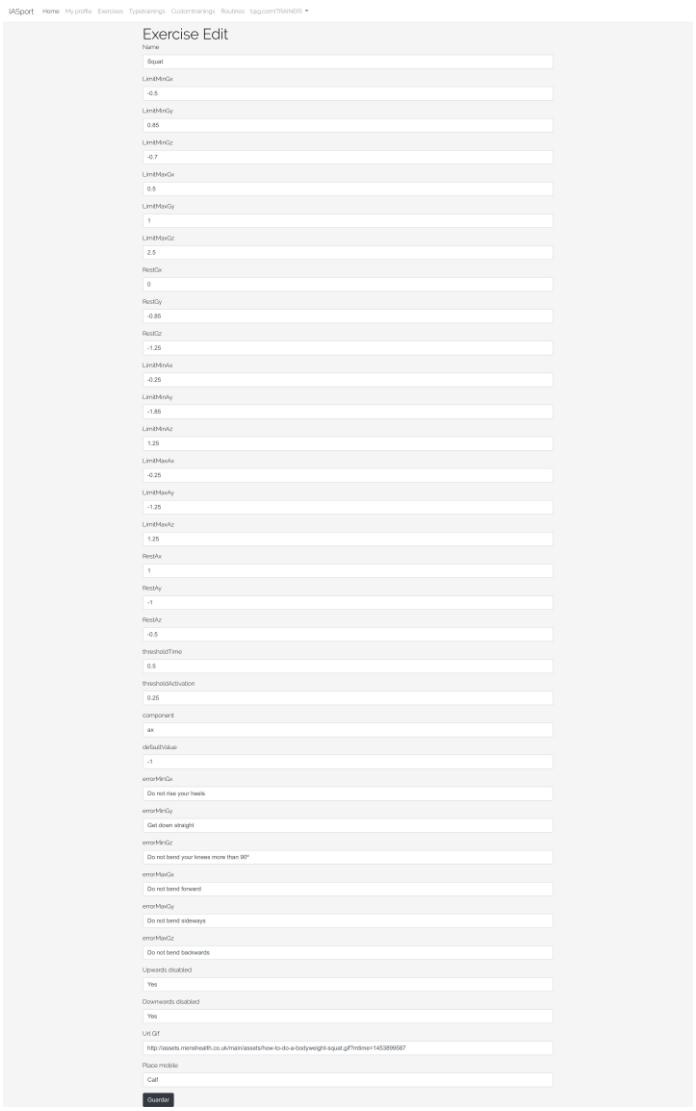


Ilustración 79 - Formulario editar ejercicio en web

8.2.2.3 Ver detalle ejercicio (trainer)

Un ejercicio, tal y como hemos comentado anteriormente, está compuesto por 33 parámetros. Esta información junto con los tipos de entrenamiento con los que está asociado pueden ser visualizados desde la web de la siguiente manera:

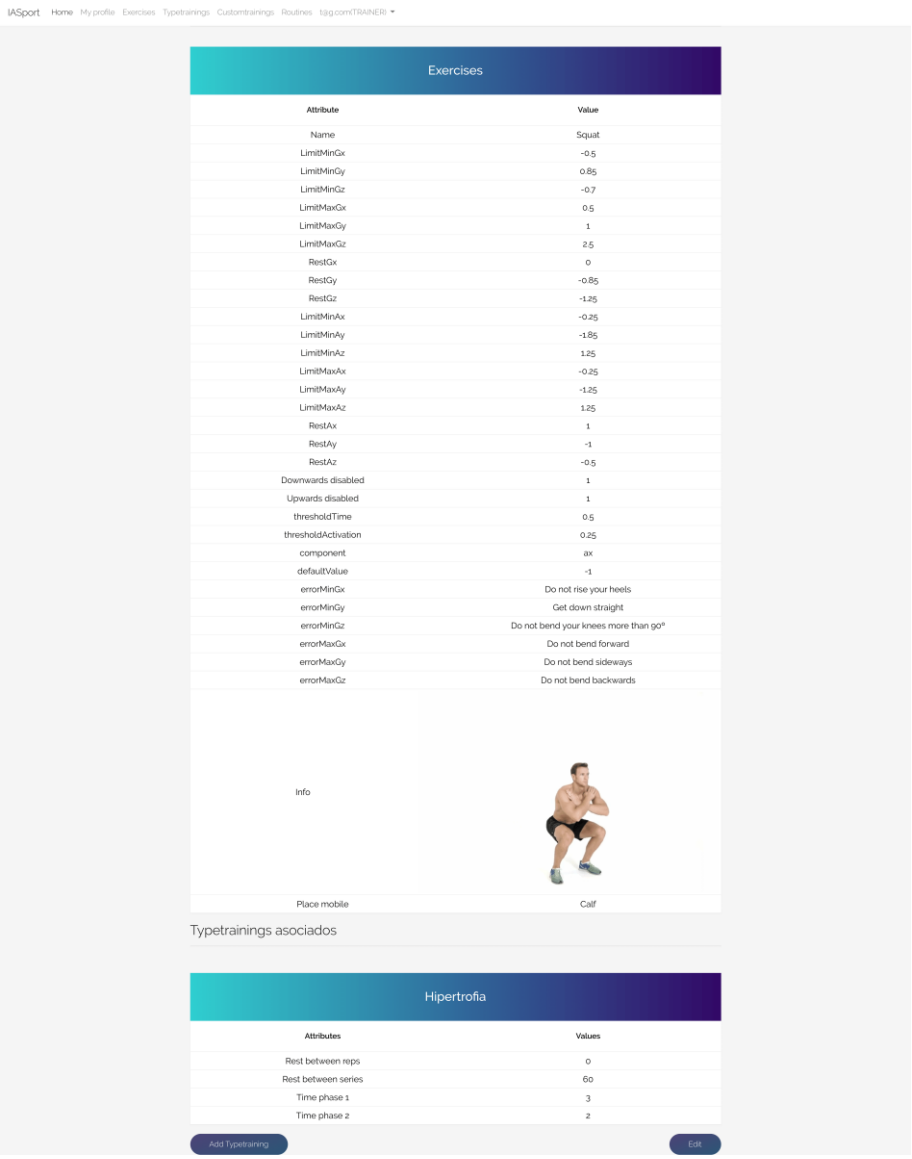


Ilustración 80 - Detalle ejercicio en web

Cabe destacar que cada vez que visitamos el detalle de un ejercicio, podemos ver también los parámetros del tipo de entrenamiento.

8.2.2.4 Añadir tipo de entrenamiento (trainer)

Un ejercicio se puede realizar siguiendo numerosos tipos de entrenamiento. Por ejemplo, podemos realizar flexiones de pecho en hipertrofia o en fuerza.

Cuando un tipo de entrenamiento se une a un ejercicio, implica una serie de parámetros extra que también se deberán tener en cuenta a la hora del entrenamiento:

Ilustración 81 - Añadir tipo de entrenamiento a un ejercicio en web

Tal y como se puede apreciar, se añaden los siguientes parámetros:

- Descanso entre repeticiones
- Descanso entre series
- Tiempo de la fase 1
- Tiempo de la fase 2

Cabe destacar que esta funcionalidad está restringida a trainers.

8.2.3 Tipo de entrenamiento

Tal y como se ha mencionado anteriormente, cada ejercicio puede estar asociado a varios tipos de entrenamiento, lo que añade parámetros extra que se deben tener en cuenta en la monitorización del mismo.

Actualmente existen una cantidad de tipos de entrenamientos que han sido creados por expertos en la materia y que la plataforma ya alberga. No obstante, la plataforma desea que este aspecto

también sea flexible, permitiendo a los trainers crear y editar tipos de entrenamiento según crean conveniente.

8.2.3.1 Crear tipo de entrenamiento (trainer)

Para crear un tipo de entrenamiento, basta con rellenar el formulario pensado para ello, otorgándole un nombre.

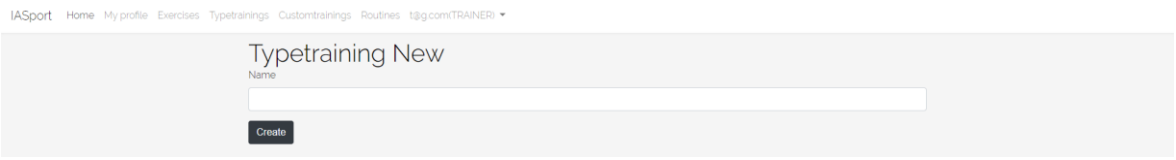
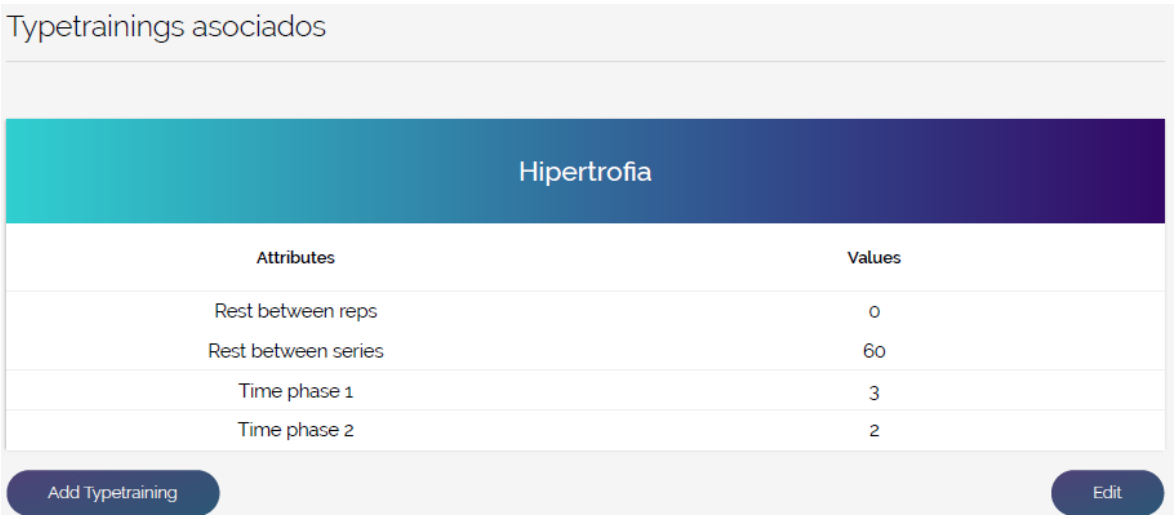


Ilustración 82 - Formulario crear tipo de entrenamiento en web

8.2.3.2 Editar tipo de entrenamiento (trainer)

El tipo de entrenamiento es un nombre que identifica al mismo. La edición del mismo se entiende como la posibilidad de editar cada uno de los parámetros que éste ha añadido a un ejercicio (ya que depende de éste).

Tal y como hemos visto en el apartado de visualizar detalle de un ejercicio, al final aparecen todos y cada uno de los tipos de entrenamiento que se le han asociado:



Si pulsamos sobre el botón editar, podemos cambiar los parámetros que el tipo de entrenamiento “Hipertrofia” añade al ejercicio “Squat”:

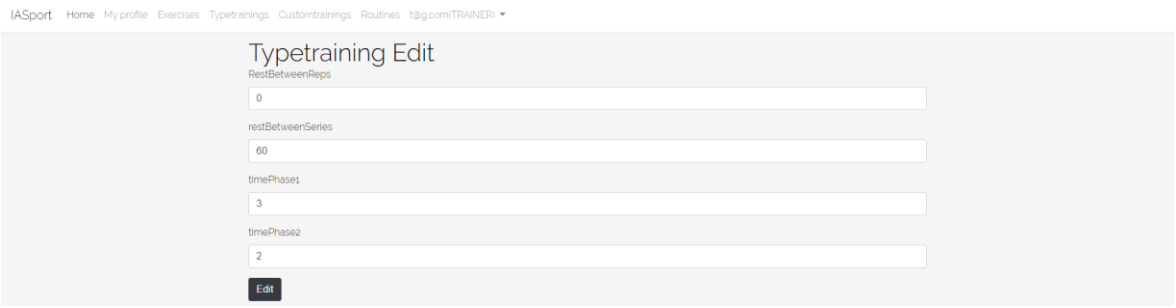


Ilustración 83 - Formulario edición tipo de entrenamiento en web

8.2.4 Rutinas

8.2.4.1 Crear rutinas (trainer)

La creación de rutinas es la unión de varios ejercicios junto con sus tipos de entrenamientos. Cada uno de estos ejercicios se debe realizar un número de series un número de repeticiones. Esta información junto con el orden en el que cada ejercicio debe ser realizado, el nombre de la rutina y el número total de días que se debe realizar es la necesaria para dar de alta una nueva rutina:

Ilustración 84 - Formulario crear nueva rutina en web

A medida que el trainer presiona el botón “Add exercise”, aparecerán de nuevo los cambios “Exercise”, “Order”, “Series” y “Reps” para añadir tantos como considere.

8.3 Funcionalidades aplicación móvil iOS

8.3.1 Entrenar (usuario normal)

Train es la vista donde se realizan todos y cada uno de los ejercicios de una rutina. Tal y como se ha mencionado anteriormente, es accedida al pulsar sobre cualquiera de los ejercicios de una rutina a la que se está apuntado. El diseño de esta vista debe ser minimalista, mostrando única y exclusivamente los datos mínimos que un usuario necesita para poder realizar el ejercicio a corde a los parámetros establecidos.

La vista es de la siguiente manera:

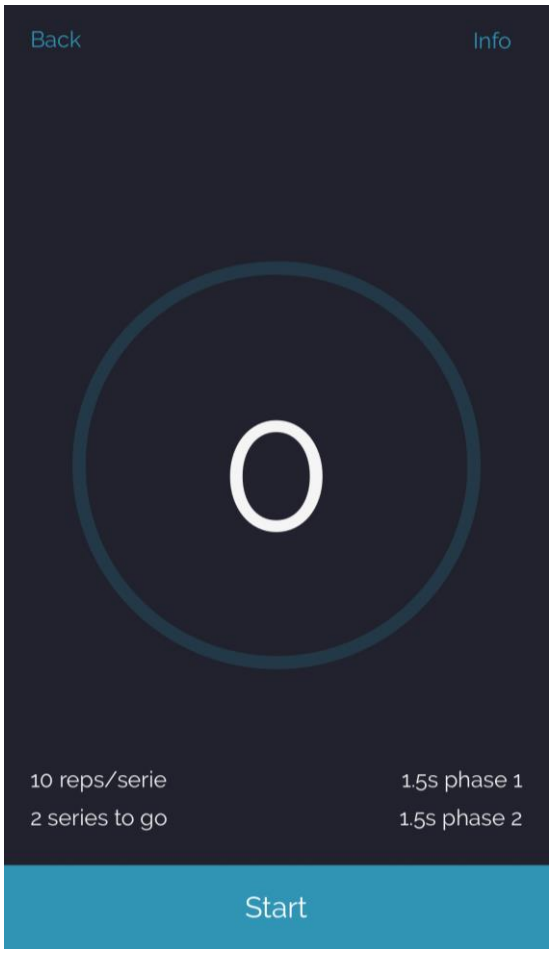


Ilustración 85 – Vista train en iOS

La información mostrada es: las repeticiones que se deben realizar en cada serie (“10 reps/serie”), el número de series que faltan para dar el ejercicio por terminado en el día actual (“2 series to go”), el tiempo de flexión (“1.5s phase 1”), así como también el tiempo de extensión (“1.5s phase 2”).

Asimismo, se muestra un círculo (el cuál se pensó que fuera dibujado con una especie de animación) y dentro el número de repeticiones realizadas hasta ahora.

Toda esta información es solicitada a la API tan pronto como la vista se carga. Sin estos parámetros, la monitorización del ejercicio es impensable, por lo que en caso de que suceda algún error, éste debe ser mostrado y no se le debe permitir al usuario comenzar el ejercicio.

La vista con el mensaje quedaría de la siguiente manera:

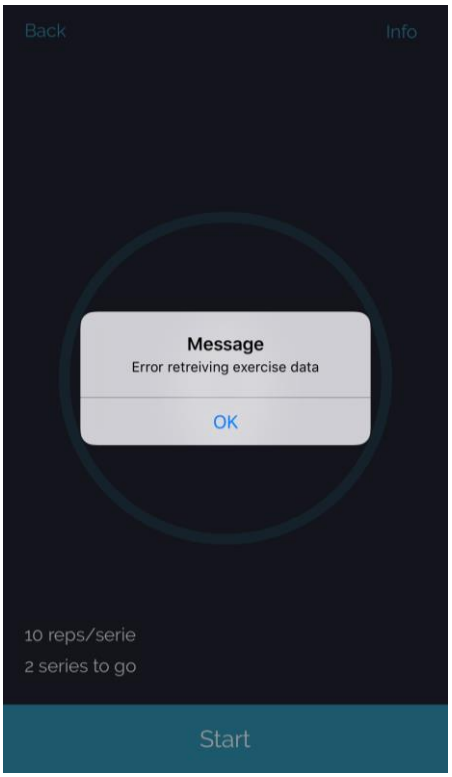


Ilustración 86 – Vista Train con mensaje de error recogiendo parámetros de la API en iOS

El mensaje puede contener información de índole diversa, pues está pensado para notificar al usuario cuando:

- Haya ocurrido un error de conexión.

- Haya realizado un error en la realización del ejercicio (en cuyo caso especificaría cual ha sido el error cometido).
- Haya terminado satisfactoriamente todas las repeticiones de la serie.

Una vez el usuario comienza el entrenamiento, donde aparece el “0” se realizará una cuenta atrás (“3”, “2”, “1” y “0”). Todos los avisos se notificarán tanto de forma visual como de forma auditiva para una mayor comodidad en la realización del ejercicio.

El usuario realizará las sucesivas repeticiones que serán monitorizadas por el algoritmo IASport. Éstas serán anotadas en la interfaz en el círculo central de la siguiente manera:

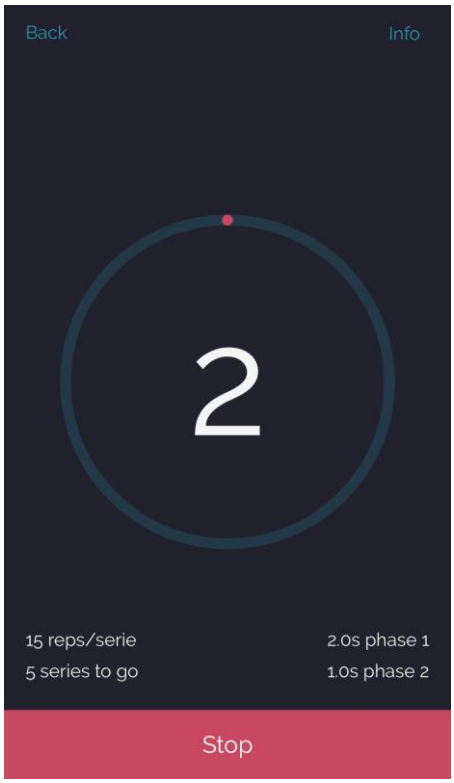


Ilustración 87 - Vista Train con entrenamiento empezado en iOS

En este caso, ya hemos realizado 2 repeticiones.

Tal y como hemos comentado anteriormente, el sistema de monitorización es encargado de comprobar que tanto la técnica como los tiempos de la realización del ejercicio concuerdan con los parámetros establecidos. En caso de que el usuario cometa un error, éste será notificado:

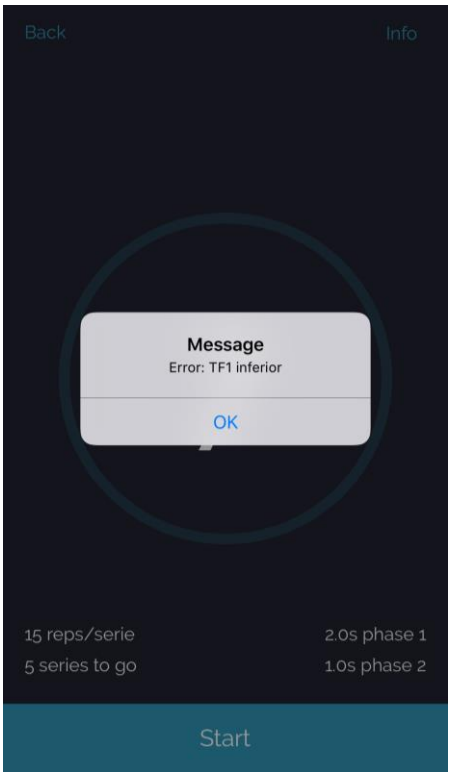


Ilustración 88 - Vista error entrenamiento en iOS

Tras haber cometido un error o haber terminado la serie, se enviará un informe a la API para las estadísticas y progresión del usuario con lo ocurrido. Posteriormente, comprueba si se han realizado el número de series. En caso afirmativo, da por terminado el entrenamiento. En caso de que todavía queden series por hacer, hará una cuenta atrás del descanso entre series que debe realizar y, posteriormente, iniciará automáticamente de nuevo el entrenamiento de la nueva serie.

8.3.2 Visualizar información ejercicio (usuario normal)

Otra información crucial es la de responder a las preguntas cómo debo realizar el ejercicio y donde debo colocar el móvil. Es por ello que se colocó un botón en la parte superior derecha de la vista "Train" que haría que se mostrara un "pop up" con dicha información.

El resultado fue el siguiente:

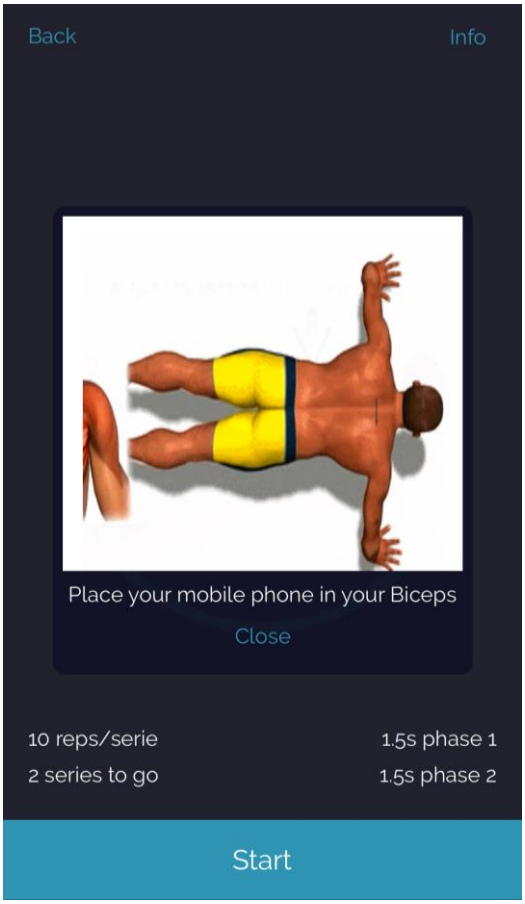


Ilustración 89 - Vista información ejercicio en iOS

En el “pop up” tendremos un gif proveniente de nuestra CDN donde se mostrará la correcta realización del ejercicio, un texto que detallará el lugar de colocación del dispositivo y un botón para cerrar el “pop up”.

8.3.3 Entrenar custom training (trainer)

Tal y como hemos mencionado anteriormente, la flexibilidad en la creación de nuevos ejercicios es un objetivo crucial para IASport. Es por ello por lo que se le debe permitir al trainer realizar movimientos que serán visualizados posteriormente en la web.

El diseño es el siguiente:

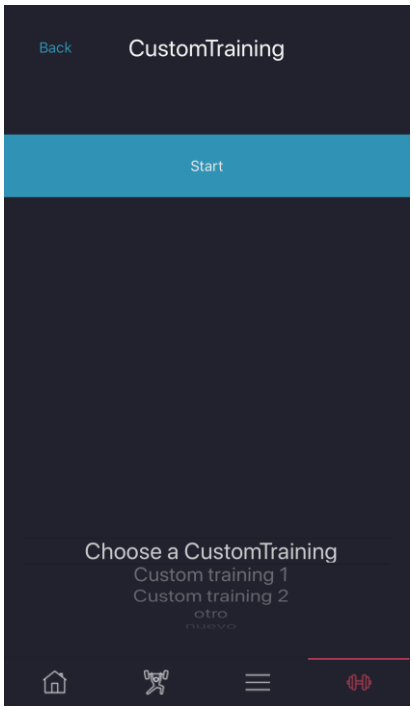


Ilustración 90 – Vista custom training en iOS

Posee un selector en el cual se puede seleccionar el custom training con el que se desea que se asocien los datos capturados. La interfaz debe tener deshabilitado el botón “Start” hasta que no se haya seleccionado uno.

Una vez el ejercicio haya comenzado, el botón de “Start” pasará a “Stop”, y el ejercicio será captado por los sensores:

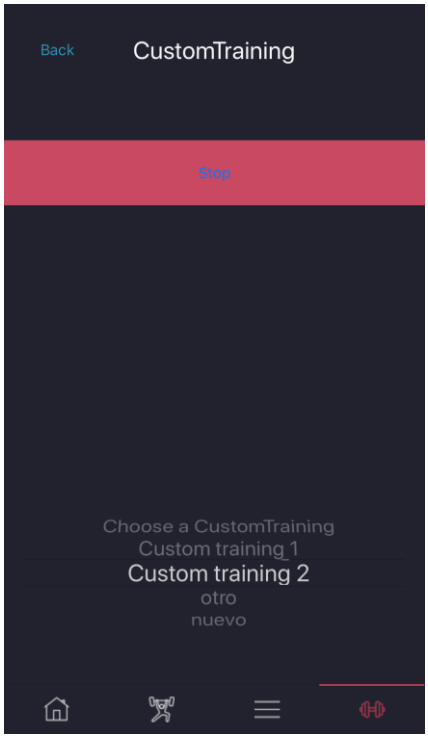


Ilustración 91 - Vista custom training captación comenzada en iOS

8.3.4 Enviar datos entrenamiento custom training

Una vez el trainer desee poner fin al ejercicio, pulsará sobre el botón “Stop”. Ya que la cantidad de datos captados puede ser muy grande, el botón de “Stop” cambiará a “Sync…” para indicar que los datos están siendo enviados a la API. Se vería de la siguiente manera:

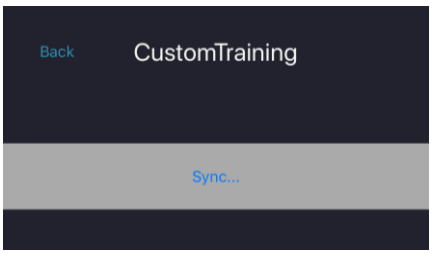


Ilustración 92 - Vista custom training sincronizando datos en iOS

Al finalizar, el botón pasará de nuevo a “Start” y todos los datos habrán sido sincronizados.

8.3.5 Avisos de inicio rutina (usuario normal)

Tal y como hemos comentado anteriormente, una vez un usuario decide apuntarse a una nueva rutina, puede decidir los días y las horas a las que desea realizarla. El cliente móvil iOS tendrá en cuenta esta información y mostrará notificaciones cuando le toque entrenar a modo de recordatorio.

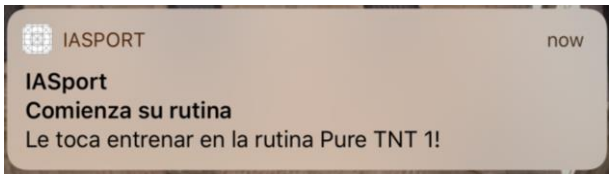


Ilustración 93 - Notificación entrenar iOS

9. Seguridad

IASport hace uso de diversas medidas de seguridad para asegurar que los datos del usuario se encuentran almacenados de forma segura.

9.1 Contraseñas

La contraseña de un usuario es la pieza clave en la seguridad del sistema, pues le otorga acceso a este. Por ello, se han llevado a cabo diversas medidas para garantizar que se almacenan de forma segura.

El sistema, en cuanto a contraseñas se refiere, funciona como un servidor de conocimiento cero, es decir, el servidor es incapaz de conocer la contraseña de sus usuarios. Desde los clientes (web e iOS), la contraseña es resumida con una función hash SHA512 y es enviada únicamente su mitad menos significativa (la mitad de la derecha). Además, una vez llega al servidor, esta es hasheada de nuevo con bcrypt (gracias al paquete org.mindrot.jbcrypt). La aplicación de bcrypt en el servidor evita que se puedan realizar ataques del tipo “tablas de arcoíris”, así como también evita que se pueda utilizar el paralelismo de las GPU’s para romper la contraseña.

Gracias a estas medidas de seguridad, evita que los ataques del tipo hombre de en medio (man-in-the-middle) puedan conocer la contraseña del usuario en su totalidad. Además de estas medidas, la comunicación viaja cifrada con un certificado HTTPS/TLS, cortesía de let’s encrypt.

9.2 Roles

Actualmente el sistema posee tres roles de usuarios claramente definidos:

- Usuario no identificado: usuario que puede registrarse y autenticarse en el sistema.
- Usuario normal: usuario convencional el cual puede apuntarse a rutinas, ver la lista de rutinas, entrenar o ver sus estadísticas.
- “Trainer”: usuario encargado de la administración del sistema. Puede llevar a cabo las tareas de crear rutinas, ejercicios, tipos de entrenamiento, así como también nuevos ejercicios customizados. Cabe destacar que el trainer a su vez también es un usuario normal (un trainer está compuesto por un usuario).

En cada una de las peticiones que llegan a la API (y que requieran de autenticación), debe existir un token válido (asociado a un usuario y no caducado). Una vez dicho token es validado, se comprueba si el usuario posee el rol necesario para llevar a cabo la tarea. En la siguiente ilustración podemos ver como comprobamos el token para la tarea de crear un nuevo custom training, el cual precisa que el usuario sea "trainer":

```
@RequestMapping(path="/short/{id}", method = RequestMethod.GET)
public @ResponseBody CustomTrainingShortResponse getCustomTrainingShort(@RequestHeader("Authorization") String authHeader,
        CustomTrainingShortResponse customTrainingShortResponse = new CustomTrainingShortResponse();
        try {
                System.out.println("1");
                String token = authHeader.split(" ")[1];
                Authtoken a = authtokenRepository.findByToken(token);
                User u = a.getUser();
                Trainer t = trainerRepository.findByUser(u);
                if(t != null) {
```

Ilustración 94 - Comprobación rol para crear custom training

9.3 JPA

JPA, tal y como se ha explicado en el apartado de tecnología, abstrae el acceso a la base de datos mediante el uso de objetos POJO. Esto hace que se utilicen consultas directas, evitando ataques como SQL injection.

Asimismo, el lenguaje de consultas que pose JPA, llamado JPQL, hace uso de los objetos de entidad de JPA, y no con las tablas de las bases de datos como hace SQL, lo que protege al sistema de SQL injection todavía más.

9.4 JWT

Estándar basado en JSON encargado de la creación de tokens con una serie de claims. El sistema leería el token que se encuentra en la cabecera "Authorization" que va precedido de la palabra reservada "Bearer" y, en función de éste, se le otorgarán unos permisos u otros.

Los JWT están compuestos de header (contiene el algoritmo utilizado y el tipo de token), payload (alberga los distintos claims) y signature (posee la concatenación de header y payload separados de "." y encriptados con HMAC SHA256). Las tres partes están codificadas en base 64.

El JWT es obtenido mediante la unión de las tres partes seguidas de un ".".

Los JWT's son generados cuando el usuario se loga satisfactoriamente en el sistema. Veamos un ejemplo de la respuesta login:


```
{
    "id" : "8657",
    "jwt" : "eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE1MTk5MzUxMzgsInN1YiI6InVzZXJzL1R6TVVvY01GNHAiLCJleHAiOjE1MTk5MzUxNDF9.9xJKw0BHjUID6-n_3bPx3PPdPS7QN7J6UsJIx2Xu0cLqLqcrd1H8vF-aVvP9sF7PMUqUjgj9dStIyunzXagtzw",
    "rol" : "NORMAL"
}
```


Ilustración 95 - Ejemplo respuesta login JSON

9.5 Web y aplicación móvil iOS

Tal y como se ha mencionado anteriormente, la API funciona como un servidor de conocimiento cero en cuanto a contraseñas se refiere, pues sólo es enviado el hash SHA512 de la mitad menos significativa desde los dos clientes actualmente desarrollados.

Además, éstos no mantienen una sesión independiente de la API, si no que la sesión se mantiene centralizada a través del JWT generado. Los clientes almacenan el token recibido desde la API para enviarlo a través de la cabecera "Authorization" precedido de la palabra "Bearer" cuando la petición lo requiera.

```
POST /exercise HTTP/1.1
Host: iasport.es
Content-Type: application/json
Authorization: Bearer eyJhbGciOiJIUzUxMiJ9
    .eyJpYXQiOjE1MTk5MzUxMzgsInN1YiI6InVzZXJzL1R6TVVvY01GNHAiLCJleHAiOjE1MTk5MzUxNDF9
    .9xJKw0BHjUID6-n_3bPx3PPdPS7QN7J6UsJIx2Xu0cLqLqcrd1H8vF-aVvP9sF7PMUqUjgj9dStIyunzXagtzw
Cache-Control: no-cache
{
    ...
}
```

Ilustración 96 - Ejemplo petición desde cliente web o iOS

9.6 Certificados SSL

Además de toda la seguridad anteriormente mencionada, la información viaja cifrada gracias a los certificados SSL cortesía de let's encrypt.

Existen dos certificados, uno instalado en el apache encargado de servir la web y otro en el apache Tomcat donde se encuentra la API. Esto garantiza una transferencia segura de datos tanto a los navegadores como al cliente móvil iOS.

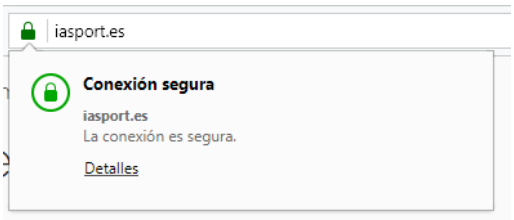


Ilustración 97 - HTTPS iasport.es

10. Testing

10.1 Tests de aceptación

Se realizaron algunos tests de aceptación en la web a través de la herramienta Katalon, garantizando así que el conjunto de comportamientos programador coincide con los especificados:

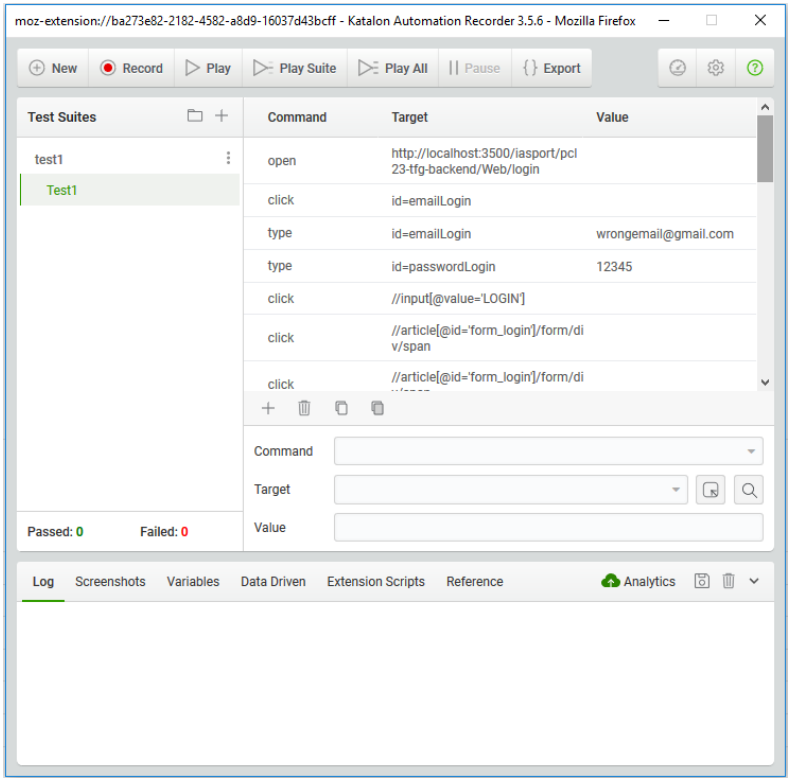


Ilustración 98 - Ejemplo test Katalon

10.2 Tests de usabilidad

Se han realizado pruebas de usabilidad en el que varios usuarios reales han probado la aplicación en su propio móvil. Tras las pruebas, se detectaron algunos errores de usabilidad. El más significativo fue la imposibilidad de varios usuarios de encontrar la vista Train, pues no existía nada que les hiciera pensar que, tras pulsar el ejercicio, iban a acceder a dicha vista. Se solucionó añadiendo una flecha.

10.3 Postman

Por último, Postman permite realizar suites de tests y ejecutarlos automáticamente siempre que se desee. Estos tests son completamente mantenibles y, sobretodo, repetibles.

Se encuentran almacenados en formato json en la carpeta raíz del proyecto.

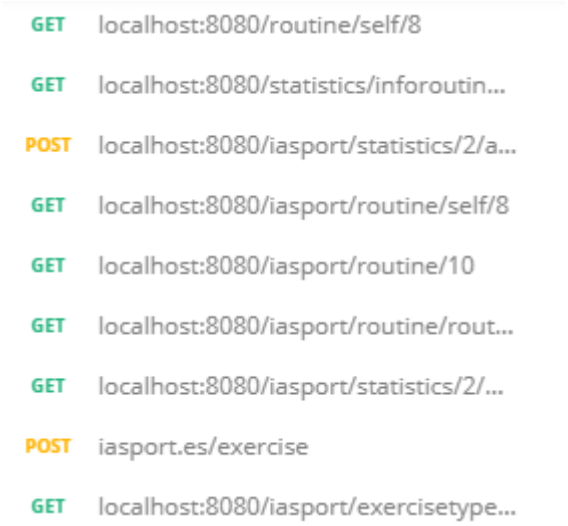


Ilustración 99 - Ejemplo urls probadas con Postman

11. Repositorios y puesta en marcha

A continuación, se encontrará la lista de los repositorios, así como también la puesta en marcha del proyecto completo de IASport.

11.1 Repositorios

Tal y como se ha comentado anteriormente, los repositorios se encuentran alojados en GitHub:

- API REST, Web y CDN: https://github.com/pcl23ua/pcl23-tfg-backend
- Aplicación iOS: https://github.com/pcl23ua/pcl23-tfg-ios

11.2 Puesta en marcha

11.2.1 Dependencias

Las dependencias necesarias para poner en marcha el proyecto son las siguientes:

- Java 8: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- MySQL: https://dev.mysql.com/downloads/mysql/
- Apache Web Server: https://httpd.apache.org/download.cgi
- Apache Tomcat 8: https://tomcat.apache.org/download-80.cgi
- Xcode: https://developer.apple.com/xcode/
- Maven: https://maven.apache.org/guides/getting-started/windows-prerequisites.html
- Node.js: https://nodejs.org/es/download/
- Composer: https://getcomposer.org

A continuación, se relacionarán las partes de la arquitectura con las dependencias que precisa cada una:

- API REST: Maven, Java 8, Apache Tomcat 8 y MySQL
- CDN: Node.js
- Cliente móvil iOS: Xcode
- Cliente web: Composer, Apache Web Server

11.2.2 Despliegue

Para el despliegue de nuestra aplicación debemos realizar los pasos que a continuación se detallan. Cabe destacar que se ha comprado un servidor expresamente para albergar IASport, por lo que los pasos corresponden a un servidor Ubuntu 16.04 Server de 64 bits.

11.2.2.1 MySQL

El primer paso para desplegar nuestro proyecto, es iniciar el servidor de bases de datos MySQL. Una vez hecho esto, debemos crear una base de datos vacía. En nuestro caso le hemos puesto el nombre "iasport".

11.2.2.2 Apache Tomcat 8

El segundo paso es arrancar el Apache Tomcat 8 en el puerto 8080 mediante el comando:

```
$ systemctl start tomcat
```

Tras esto, el servidor Tomcat estará arrancado en http://localhost:8080. En nuestro caso, también poseemos el dominio iasport.es, por lo que sería accesible desde el exterior en https://iasport.es:8080/.

Asimismo, en nuestro caso instalamos un certificado SSL para que la información viajara cifrada.

11.2.2.3 API

Para poner en marcha la API, lo primero que debemos hacer es editar el fichero application.properties que podemos encontrar en src/main/resources del directorio. En él pondremos la ip, el puerto y el nombre de la base de datos, así como también las credenciales para acceder.

Posteriormente, deberemos generar el war que será lanzado por Apache Tomcat. Para ello, en el directorio raíz de nuestra API, ejecutamos:

```
$ mvn package
```

Una vez generado, lo debemos copiar en el directorio webapps de Apache Tomcat con el nombre iasport.war.

A partir de este momento nuestra API estará activa en la dirección http://localhost:8080/iasport y, en nuestro caso, desde el exterior desde https://iasport.es:8080/iasport.

11.2.2.4 Apache Web Server

Arrancamos Apache Web Server:

```
$ systemctl start apache2
```

El servidor estará disponible en la dirección http://localhost/ y, en nuestro caso, desde el exterior desde https://iasport.es.

Asimismo, se instaló un certificado SSL para que la información viajara.

11.2.2.5 Web

Para poner en marcha la web, debemos copiar el directorio completo en /var/www, pues es el directorio que hemos configurado en Apache Web Server.

Posteriormente, debemos configurar e instalar todas las dependencias a través de Composer mediante la ejecución del siguiente comando

```
$ composer install
```

Una vez termine de instalar las dependencias ejecutaremos:

```
$ php artisan key:generate
```

Tras este paso, se habrá generado la clave de cifrado que Laravel necesita.

La web ha sido configurada para ser servida desde la raíz del Apache Web Server, por lo que su dirección desde el exterior es https://iasport.es.

11.2.2.6 CDN

A continuación, necesitamos arrancar la CDN para que comience a servirnos los recursos. Dentro del directorio donde se encuentre, debemos ejecutar:

```
$ npm install
$ nodemon app.js
```

A partir de este momento, nuestra CDN servirá recursos en http://localhost:4500 y, en nuestro caso, desde el exterior en http://iasport.es:4500.

11.2.2.7 Cliente móvil iOS

Para desplegar el cliente móvil, basta con abrir el proyecto con Xcode, compilarlo y lanzarlo en algún dispositivo iPhone que se encuentra configurado.

12. Conclusiones y mejoras

12.1 Futuro de IASport

IASport es un producto que yo mismo he utilizado para entrenar desde el momento en el que el sistema de monitorización fue creado. Es por ello por lo que se continuará desarrollando y mejorando, para que cualquier persona que desee pueda hacer uso de la misma.

12.2 Mejoras

12.2.1 Entrenamientos con amigos

Sería muy interesante poder añadir amigos en la plataforma para poder entrenar con ellos.

Partiría de la base que ambos se apuntan a la misma rutina. Posteriormente, bastaría con extender la funcionalidad de entrenar para que almacenase en tiempo real cada una de las repeticiones que realiza de una serie en concreto cada uno (ya que ahora el informe es enviado al final). De esta manera, no sólo podrían retarse, sino que también podrían entrenar juntos, incluso sin estar en la misma zona geográfica.

Además, almacenando esta información, podríamos entrenar en horas distintas, pues la aplicación se descargaría los resultados automáticamente. Después ambos podrían saber los resultados del otro.

12.2.2 Retos con otras personas

Tal y como se ha comentado en el apartado anterior, la posibilidad de retarse con un amigo a la hora de realizar una rutina añadiría mucho valor al producto. Una vez se haya llevado a cabo, se puede pasar al siguiente nivel.

Los datos de la realización de una rutina de un usuario podrían estar a disposición de los demás (siempre y cuando lo desee). Una vez otro usuario se disponga a entrenar, puede elegir compararse con los datos del primer usuario a modo de reto y motivación personal. Por ejemplo, si algún famoso estuviera en la plataforma, sería muy interesante que los usuarios pudiesen descargarse sus resultados para retarse con sus ídolos.

12.2.3 Retos

La aplicación debería tener varios retos que completar para mantener a los usuarios motivados y activos. Los retos serían clasificados en diferentes grupos, en función de los tipos de ejercicios que normalmente realice el usuario, aunque también existirían genéricos.

Ejemplos de retos podrían ser:

- Entrena durante 30 días
- Realiza 50 flexiones en hipertrofia sin descansar ni cometer errores
- Realiza 50 sentadillas en hipertrofia sin descansar ni cometer errores
- Termina tres rutinas

12.2.4 Clases grupales

Un entrenador/a de un gimnasio podría tener monitorizados en tiempo real a todos y cada uno de los deportistas a su cargo. Las estadísticas durante y tras el entrenamiento le pueden dar una visión global de si el entrenamiento ha sido demasiado duro/flojo para los deportistas.

Asimismo, sabría en todo momento el estado de cada deportista individualmente.

Por último, al finalizar el entrenamiento sabrá la progresión que cada uno ha tenido en comparación con otros meses/días.

Esta nueva funcionalidad también puede ser aplicada a entrenadores personales.

12.3 Comercialización

Una vez estamos decididos a continuar con el proyecto, debemos definir un plan para su comercialización.

Se pensó en tener cuatro licencias:

- Licencias para deportistas:
    - Licencia gratuita: licencia que se adquiere cuando un usuario se registra en la plataforma. Esta licencia solo le permitirá apuntarse a rutinas, así como también ver la

información de las mismas. Se le otorgará un periodo de 15 días de prueba de la licencia sporty para que pueda disfrutar de sus beneficios.

  - Licencia sporty: licencia que se adquiere tras un pago único. Una vez se realice el pago, el usuario podrá disfrutar de la monitorización de los ejercicios, del entrenamiento con compañeros, etc.

- Licencia para entrenadores/gimnasios: licencias de pago mensual/anual
  - Licencia silver: licencia pensada para entrenadores personales que poseen varios deportistas a su cargo. La licencia le proporcionará una licencia trainer con acceso a la creación de rutinas. Asimismo, le permitirá monitorizar a 5 usuarios trained. Los usuarios trained disfrutan de los mismos privilegios que los sporty pero están asociados a una licencia silver. Cada trained que desee monitorizar extra se deberá abonar a parte comprando un paquete de licencias trained (que se añadirá al pago anual/mensual).
  - Licencia gold: licencia pensada para gimnasios. Permite crear tantos trainer como se requiera. Además, también permite crear custom trainings, pudiendo disfrutar de la flexibilidad de IASport.

En la imagen de a continuación se muestra un resumen con todas las licencias:

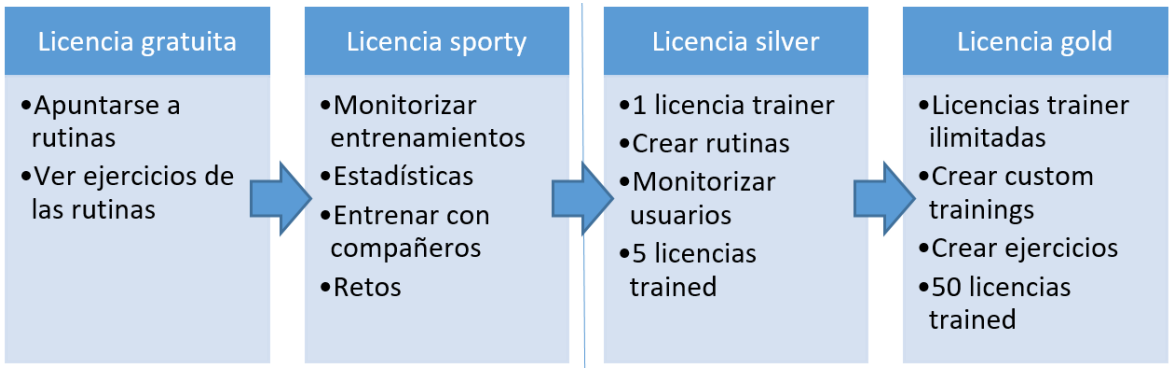


Ilustración 100 - Licencias IASport

En caso de necesitar más licencias trained, los packs son los siguientes:

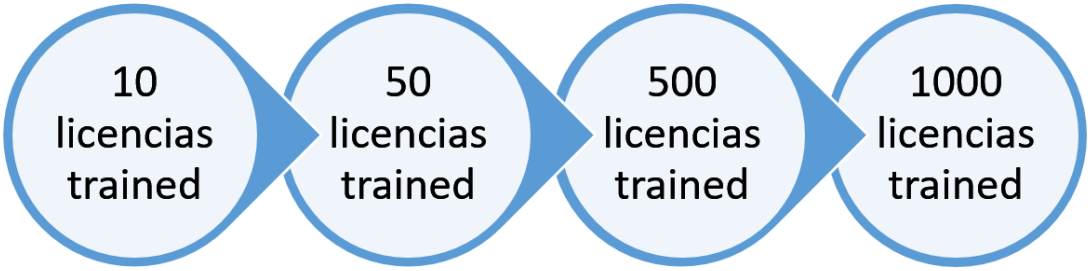


Ilustración 101 - Packs licencias trained

13. Referencias bibliográficas

1. JOVANOV, Emil; MILOSEVIC, Mladen; MILENKOVIĆ, Aleksandar. A mobile system for assessment of physiological response to posture transitions. En *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*. IEEE, 2013. p. 7205-7208.
2. CHEN, Kun-Hui, et al. Using gyroscopes and accelerometers as a practical rehabilitation monitor system after total knee arthroplasty. En *RF and Wireless Technologies for Biomedical and Healthcare Applications (IMWS-BIO), 2015 IEEE MTT-S 2015 International Microwave Workshop Series on*. IEEE, 2015. p. 58-59.
3. RASO, Iván; HERVÁS, Ramón; BRAVO, José. m-Physio: personalized accelerometer-based physical rehabilitation platform. En *Proceedings of the Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*. 2010. p. 416-421.
4. NURWANTO, Fajri; ARDIYANTO, Igi; WIBIRAMA, Sunu. Light sport exercise detection based on smartwatch and smartphone using k-Nearest Neighbor and Dynamic Time Warping algorithm. En *Information Technology and Electrical Engineering (ICITEE), 2016 8th International Conference on*. IEEE, 2016. p. 1-5.
5. RUIZ-FERNANDEZ, Daniel, et al. eFisioTrack: a telerehabilitation environment based on motion recognition using accelerometry. *The Scientific World Journal*, 2014, vol. 2014.
6. CHANG, Keng-Hao; CHEN, Mike Y.; CANNY, John. Tracking free-weight exercises. En *International Conference on Ubiquitous Computing*. Springer, Berlin, Heidelberg, 2007. p. 19-37.

- Tecnologías y herramientas utilizadas
    - Trello: https://trello.com
    - GitHub: https://github.com
    - Play Framework: https://www.playframework.com
    - Java 8: https://docs.oracle.com/javase/8/docs/api/
    - JPA: http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html
    - MySQL: https://www.mysql.com
    - JWT: https://jwt.io
    - Node.js: https://nodejs.org/es/

- Xcode: https://developer.apple.com/xcode/
- Laravel: https://laravel.com
- Lenguaje R: https://www.r-project.org
- R Studio: https://www.rstudio.com
- npm: https://www.npmjs.com
- Composer: https://getcomposer.org
- Power BI: https://powerbi.microsoft.com/es-es/
- Repositorio API REST, Web y CDN: https://github.com/pcl23ua/pcl23-tfg-backend
- Repositorio cliente móvil iOS: https://github.com/pcl23ua/pcl23-tfg-ios
- OVH: https://www.ovh.es
- URL's aplicaciones desplegadas:
  - Web: https://iasport.es
  - API REST: https://iasport.es:8080
  - CDN: http://iasport.es:4500

14. Anexos

14.1 Lista métodos API REST

- Custom Training
    - GET /customTraining
    - GET /customTraining/{id}
    - POST /customTraining
    - GET /customTraining/short/{id}
- Data Accelerometer
    - POST /dataAccelerometer
- Data Gyroscope
    - POST /dataGyroscope
- Exercise
    - POST /exercise
    - GET /exercise
    - GET /exercise/{id}
    - POST /exercise/{id}
- Exercise TypeTraining
    - GET /exercisetypetraining/{idExercise}/{idTypeTraining}
    - GET /exercisetypetraining/{id}
    - POST /exercisetypetraining
    - GET /exercisetypetraining
    - POST /exercisetypetraining/{id}
- Routine
    - GET /routine
    - POST /routine
    - GET /routine/{id}
    - GET /routine/routinesnotadded/{idUsuario}
    - GET /routine/routinesadded/{idUsuario}
    - GET /routine/self/id
    - POST /routine/{id}
- Statistics

    - GET /statistics/{idUsuario}/numberOfRepsByTypeTraining
    - GET /statistics/{idUsuario}/numberOfRepsByDate
    - GET /statistics/{idUsuario}/numberOfRepsByDateAndDays/{days}
    - GET /statistics/{idUsuario}/numberOfErrorsByType
    - GET /statistics/{idUsuario}/topnexercises/{n}
    - POST /statistics/{idUsuario}/add
    - GET /statistics/infotrainingtoday/{idUsuario}/idTraining
    - GET /statistics/inforoutinetoday/{idUsuario}/idRoutine
- Training
    - GET /training/{id}
    - POST /training/{id}
- TypeTraining
    - POST /typetraining
    - GET /typetraining
    - GET /typetraining/{id}
    - POST /typetraining/{id}
    - GET /typetraining/readallnoexercise/{id}
- User
    - POST /user/register
    - POST /user/login
- UserTraining
    - GET /usertraining/{idUsuario}
    - GET /usertraining/all/{idUsuario}
    - GET /usertraining/notactive/{idUsuario}
    - POST /usertraining/{idUsuario}/{idTraining}
    - POST /usertraining/desactivate/{idUsuario}/{idTraining}
    - GET /usertraining/{idUsuario}/{idTraining}

14.2 Lista métodos Web

- GET '/'
- GET '/login'

- POST '/login'
- POST '/register'
- GET '/logout'
- GET '/profile'
- GET '/exercises'
- GET '/exercises/{id}'
- GET '/editexercise/{id}'
- POST '/editexercise/{id}'
- GET '/newexercise'
- POST '/newexercise'
- GET '/addexercisetypetraining/{idExercise}'
- POST '/addexercisetypetraining'
- GET '/editexercisetypetraining/{id}'
- POST '/editexercisetypetraining/{id}'
- GET '/newroutine'
- POST '/newroutine
- GET '/routines'
- GET '/routines/{id}'
- GET '/myroutines/{id}'
- GET '/editroutine/{id}/{idEx}'
- POST '/editroutine/{id}'
- GET '/edittraining/{id}'
- POST '/edittraining/{id}'
- GET '/myroutines'
- GET '/addroutinetouser/{id}'
- POST '/addroutinetouser'
- GET '/desactivatemyroutine/{id}'
- GET '/typetrainings'
- GET '/typetrainings/{id}'
- GET '/edittypetraining/{id}'
- POST '/edittypetraining/{id}'
- GET '/newtypetraining'

- POST '/newtypetraining'
- GET '/customtrainings'
- GET '/newcustomtraining'
- POST '/newcustomtraining'
- GET '/customtrainings/{id}'
- GET '/raw/exercisetypetrainings'
- GET '/raw/numberOfRepsByTypeTraining/{id}'
- GET '/raw/numberOfRepsByDate/{id}'
- GET '/raw/topnexercises/{id}/{n}'
- GET '/raw/numberErrorsType/{id}'
- GET '/raw/customtrainingfull/{id}'

14.3 Mockups

14.3.1 Mockups iOS

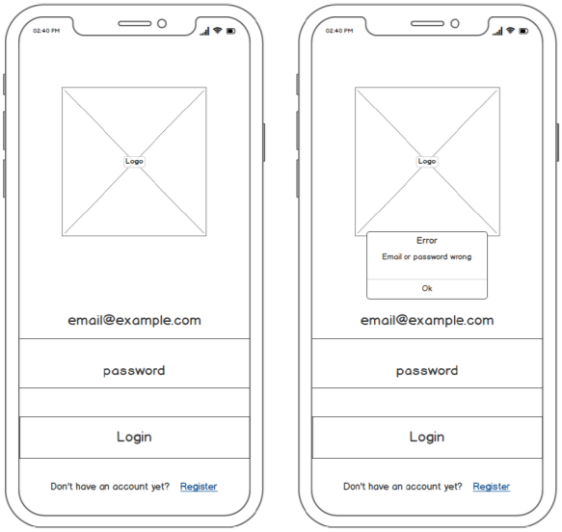


Ilustración 102 - Mockup iOS login con y sin error

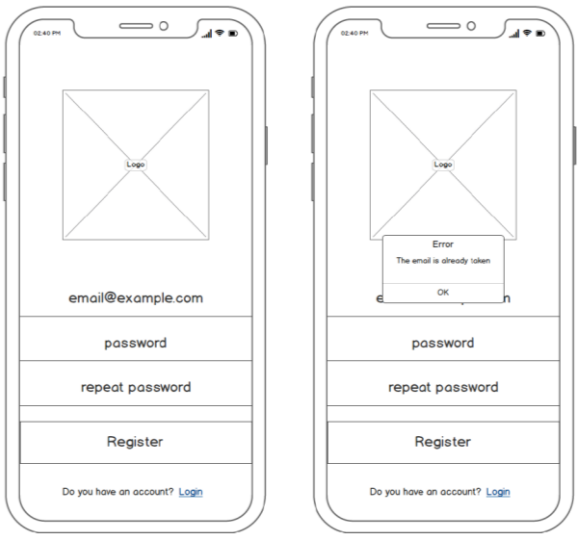


Ilustración 103 - Mockup iOS registro con y sin error

Ilustración 104 - Mockup iOS perfil usuario

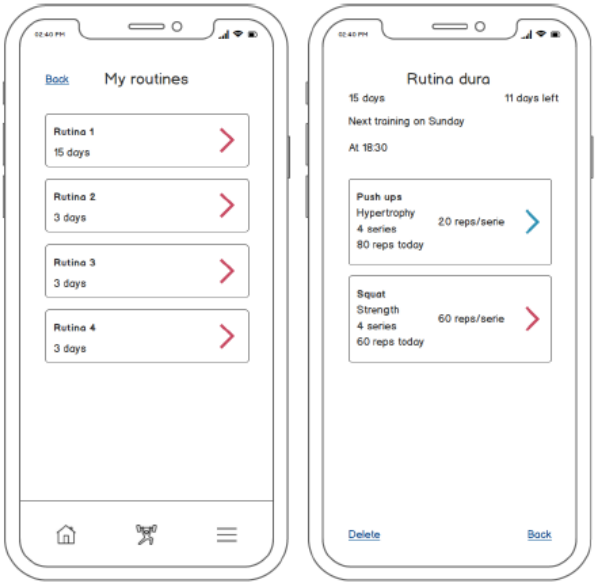


Ilustración 105 - Mockup iOS mis rutinas y detalle de rutina apuntado

Ilustración 106 - Mockup iOS Train

Ilustración 107 - Mockup iOS Train error al recibir parámetros

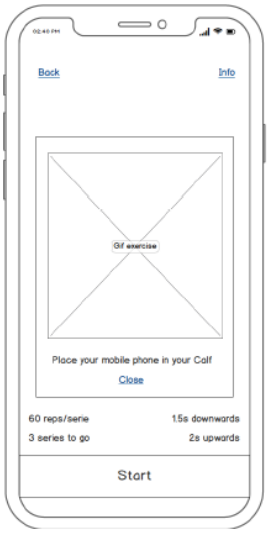


Ilustración 108 - Mockup iOS información ejercicio

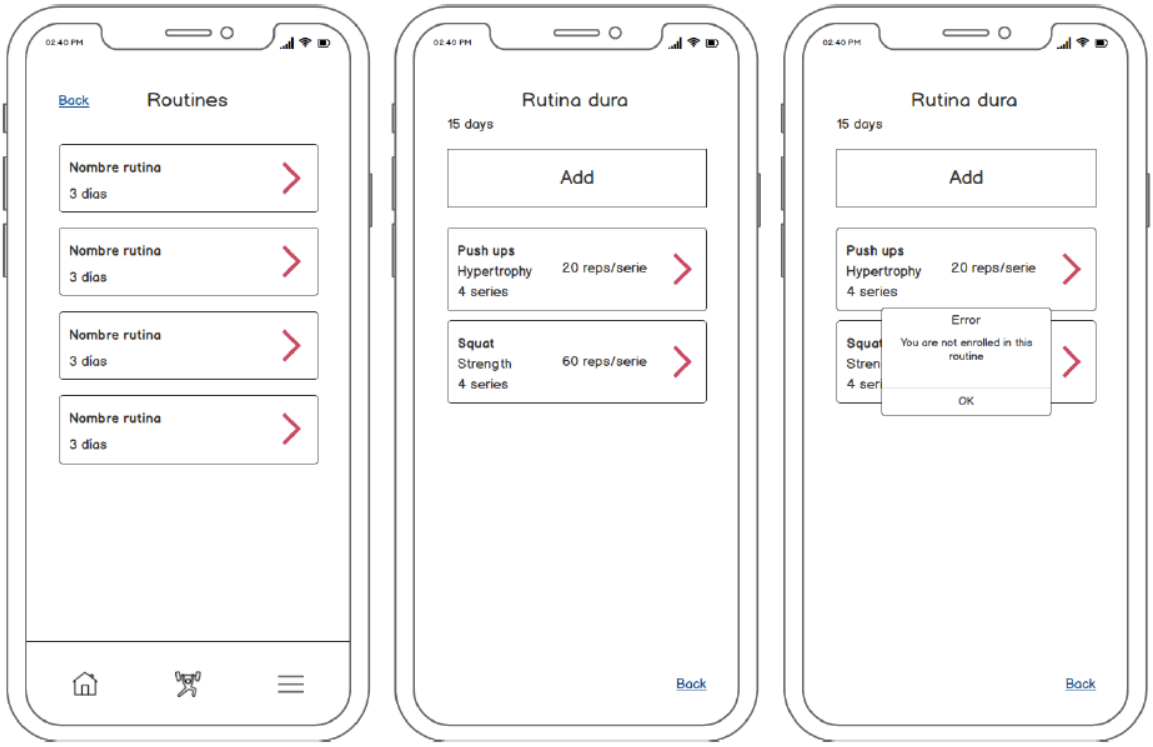


Ilustración 109 - Mockup iOS lista y detalle rutinas no apuntado

Ilustración 110 - Mockup iOS Custom training

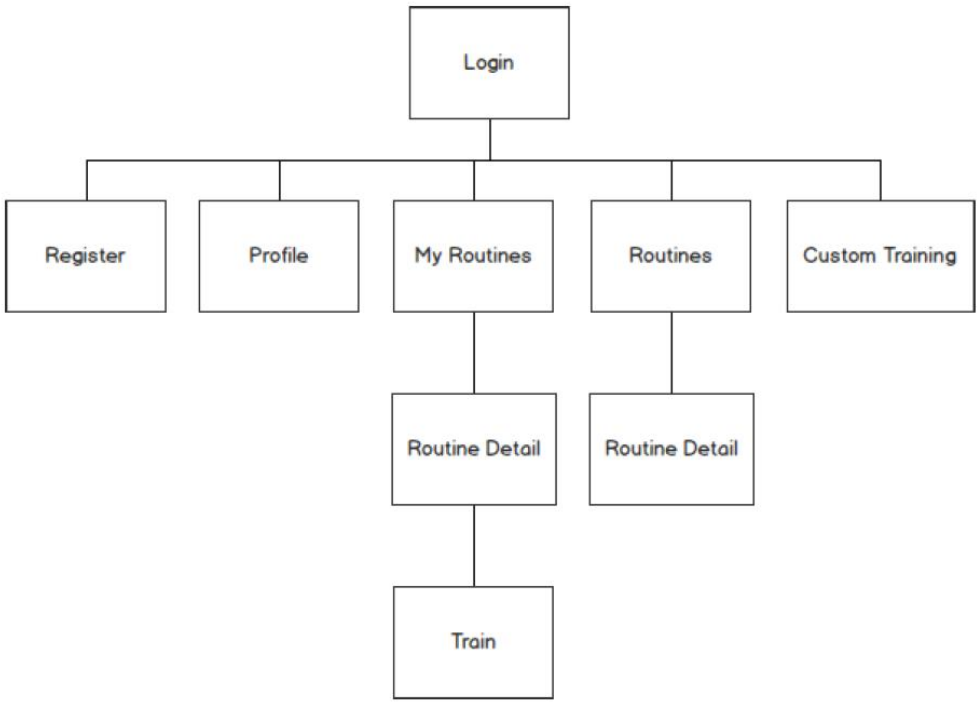


Ilustración 111 - Sitemap iOS

14.3.2 Mockups web

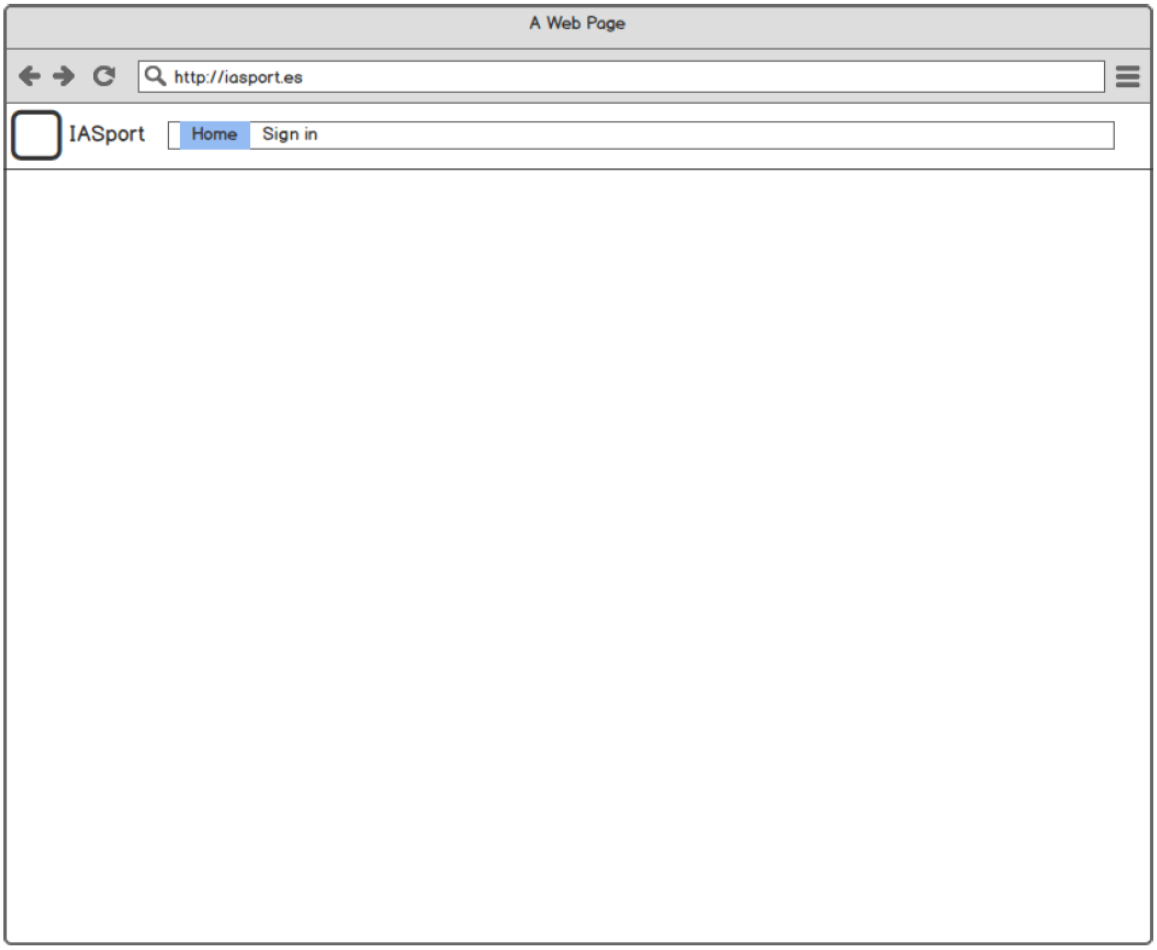


Ilustración 112 - Mockup web index

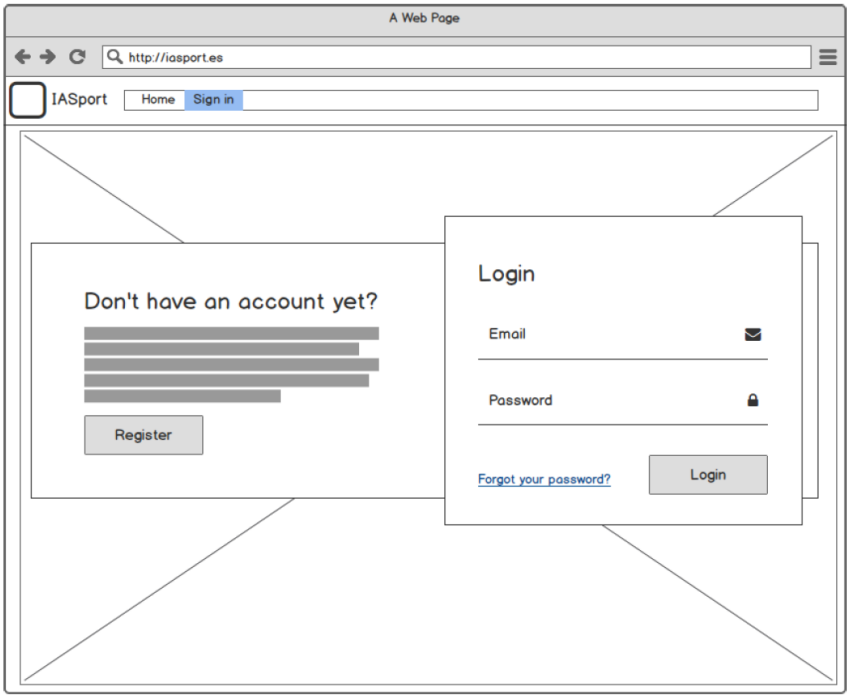


Ilustración 113 - Mockup web login

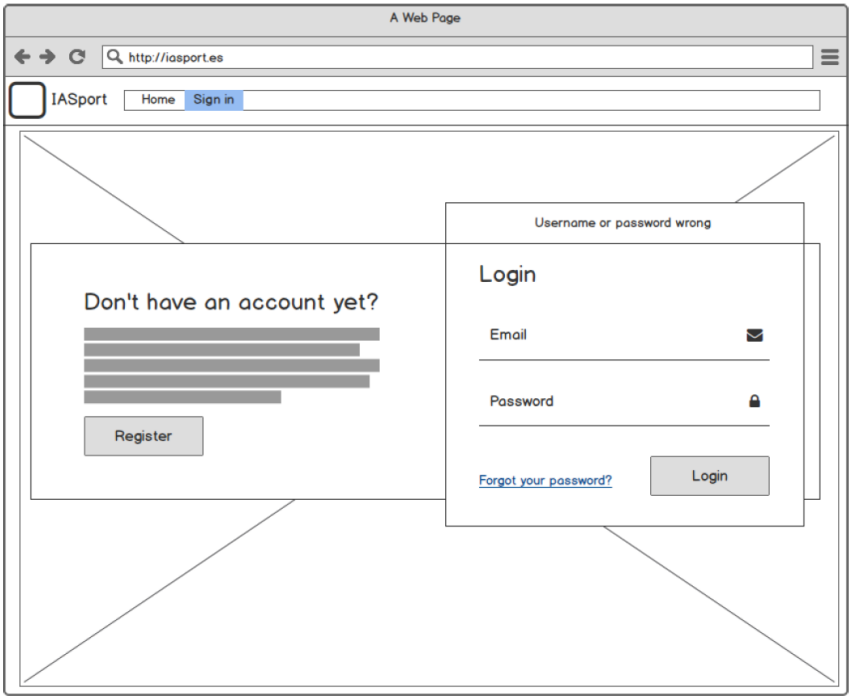


Ilustración 114 - Mockup web login con error

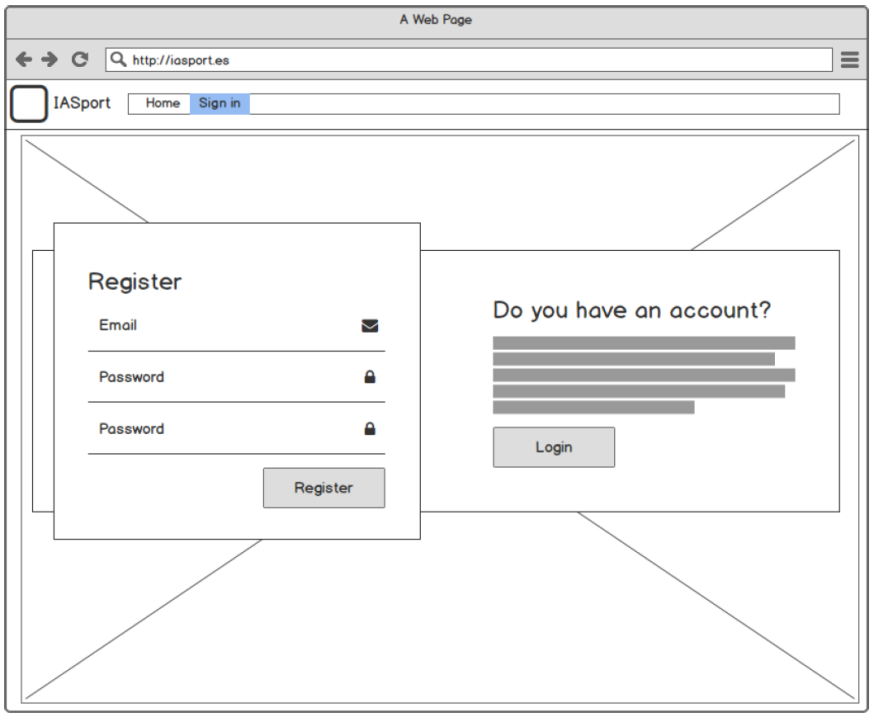


Ilustración 115 - Mockup web registro

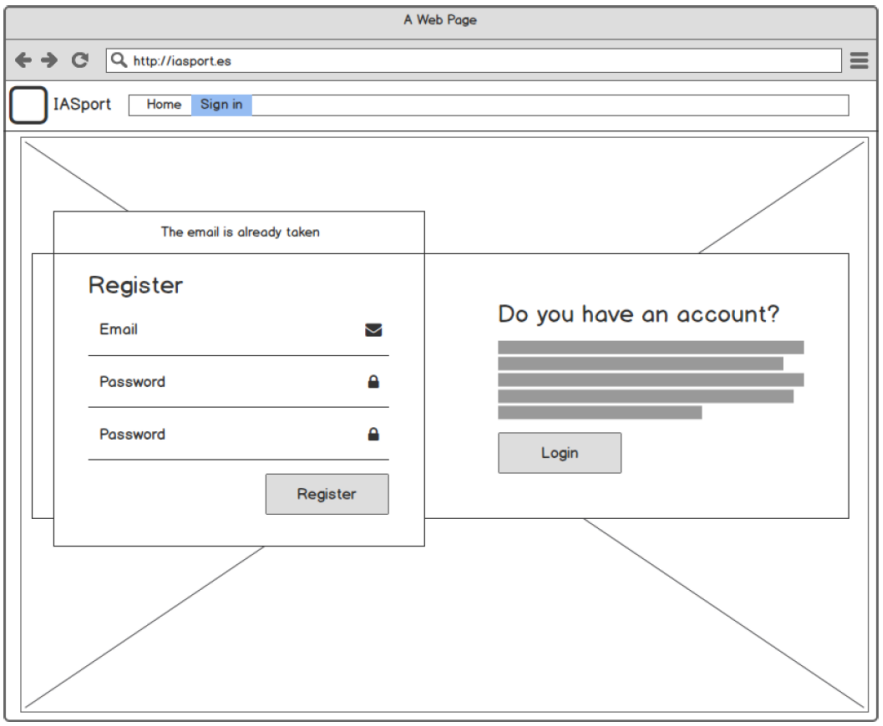


Ilustración 116 - Mockup web registro con error

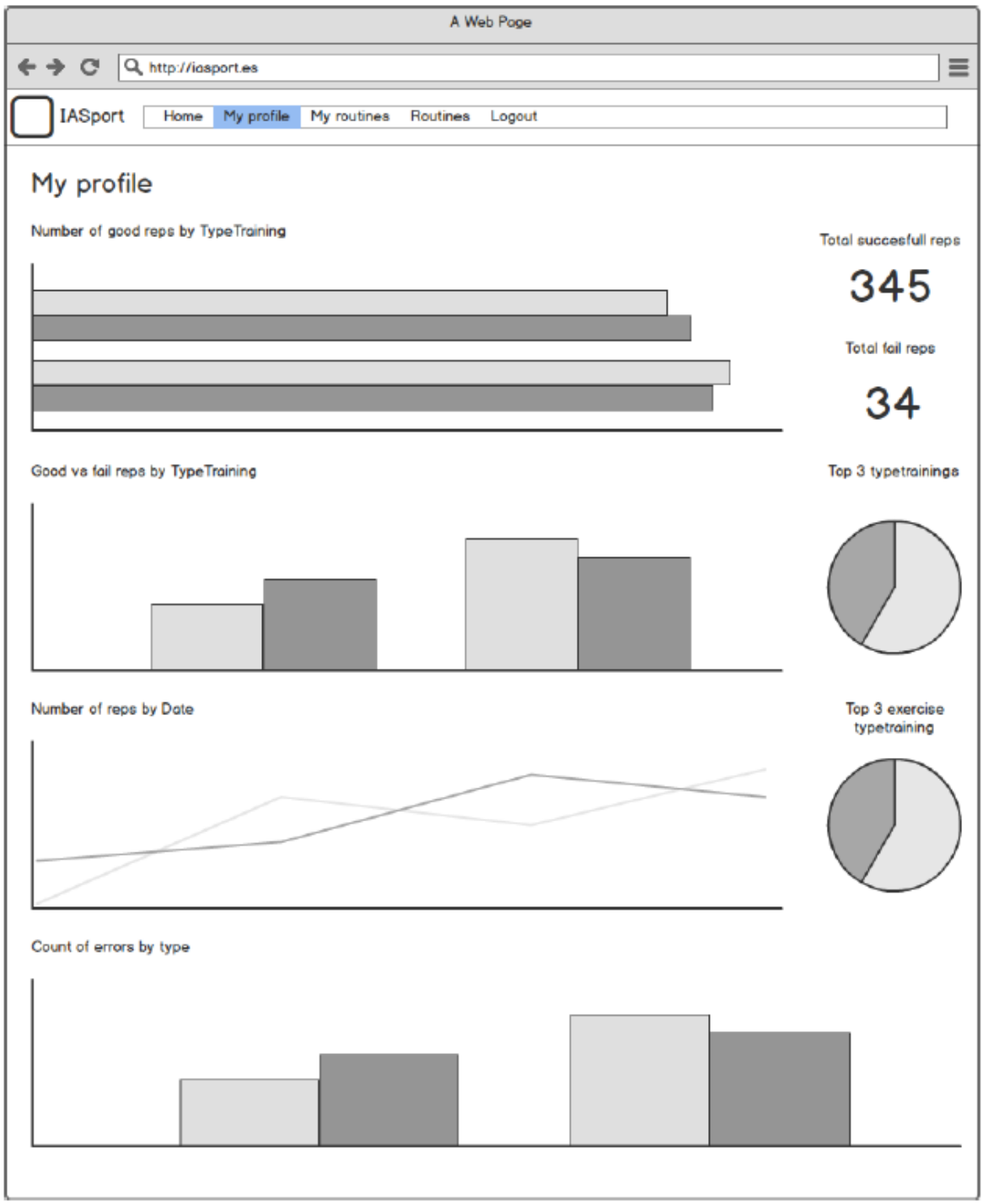


Ilustración 117 - Mockup web perfil usuario

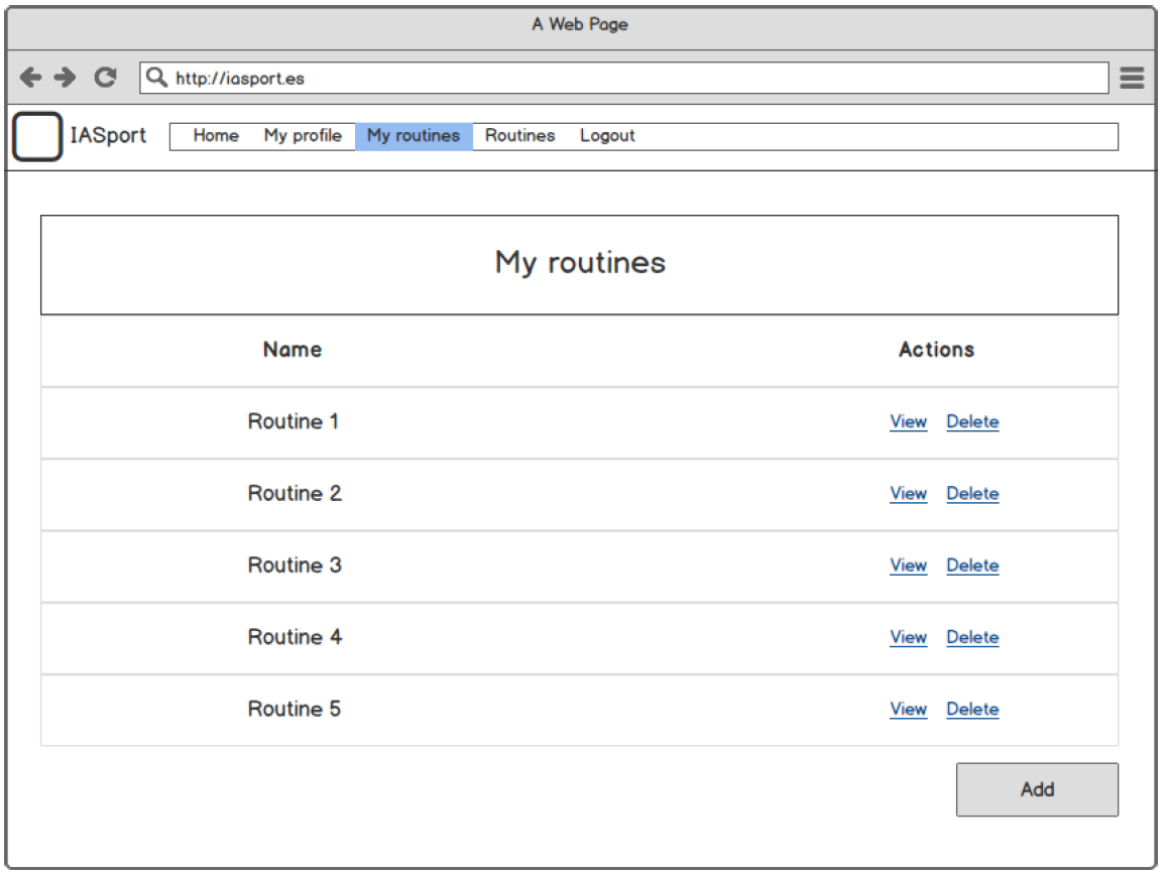


Ilustración 118 - Mockup web lista rutinas a las que se está apuntado

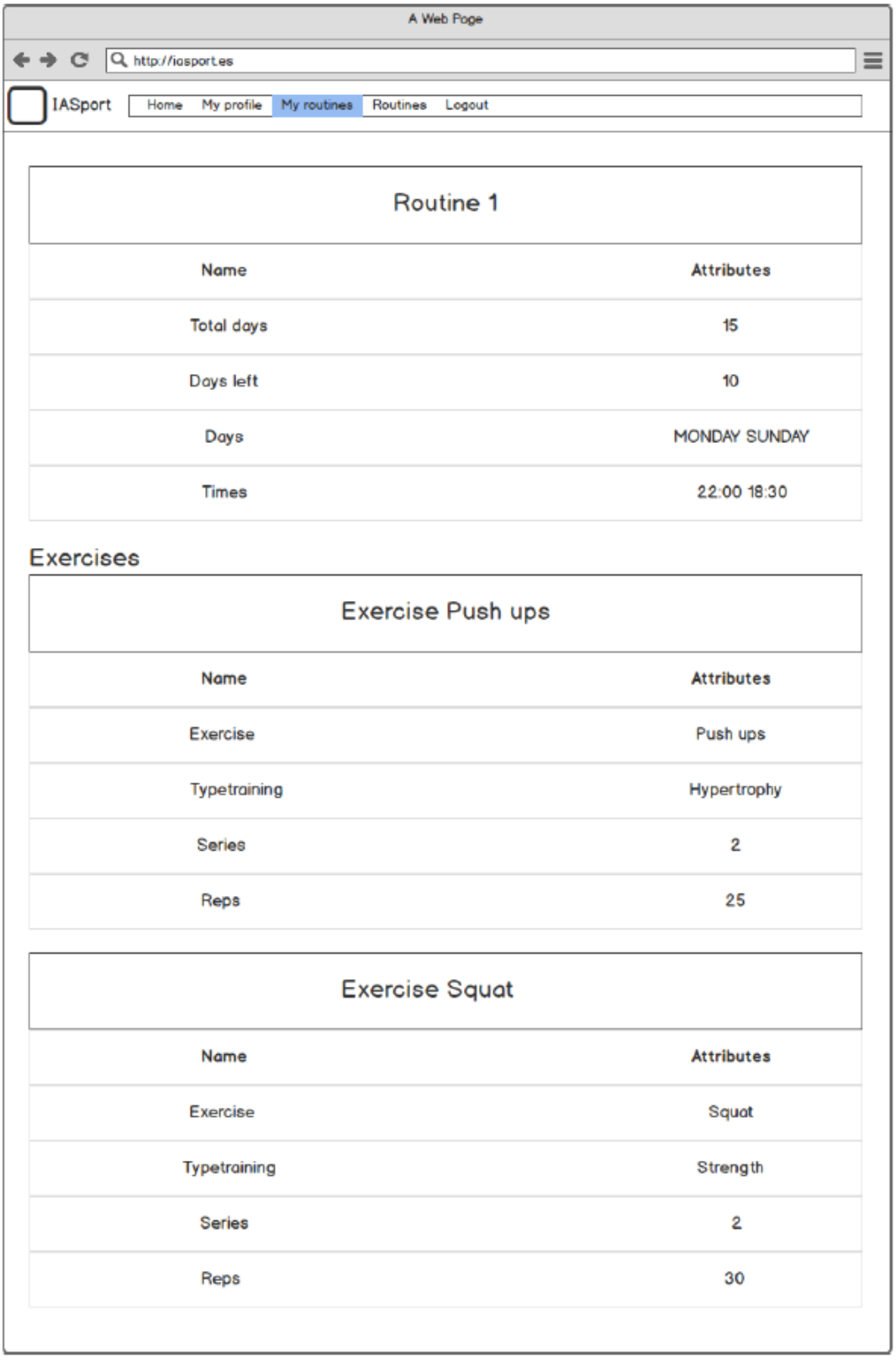


Ilustración 119 - Mockup web detalle rutina a la que se está apuntado

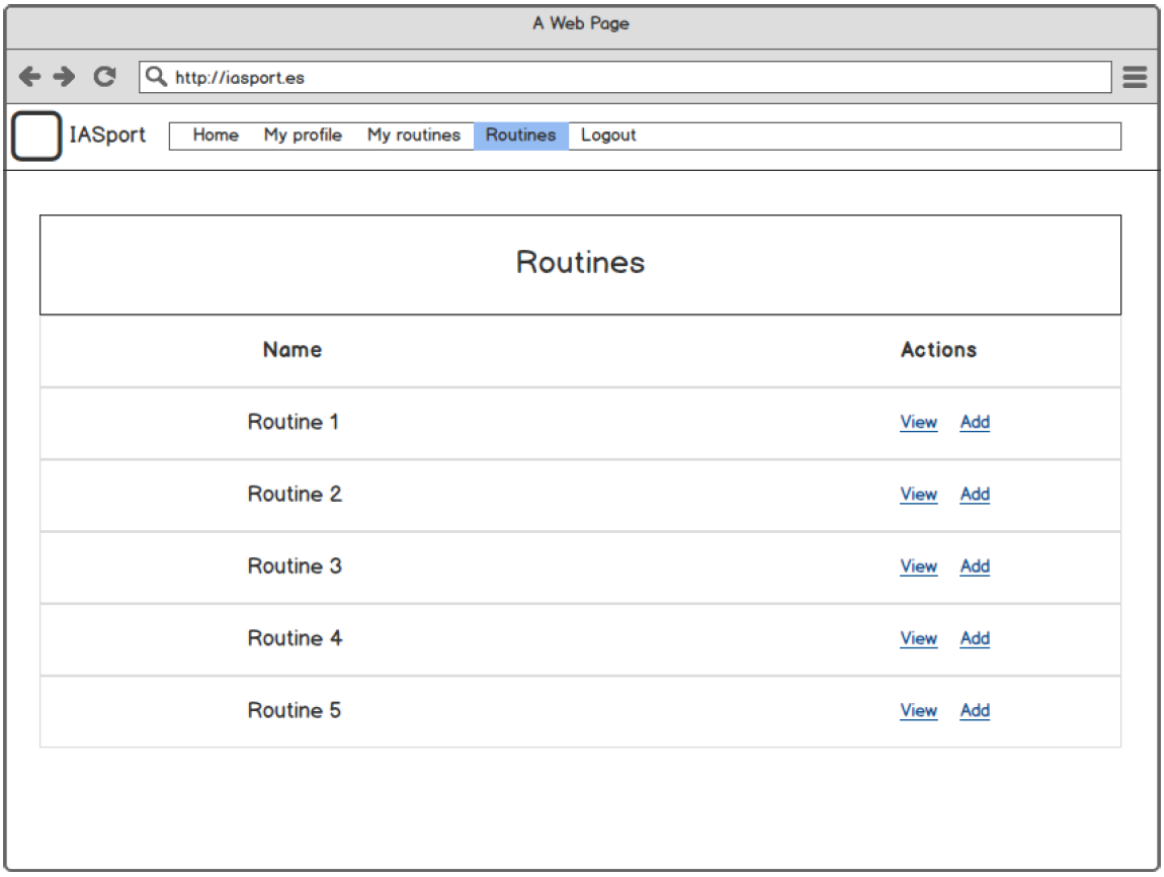


Ilustración 120 - Mockup web lista rutinas a las que no se está apuntado

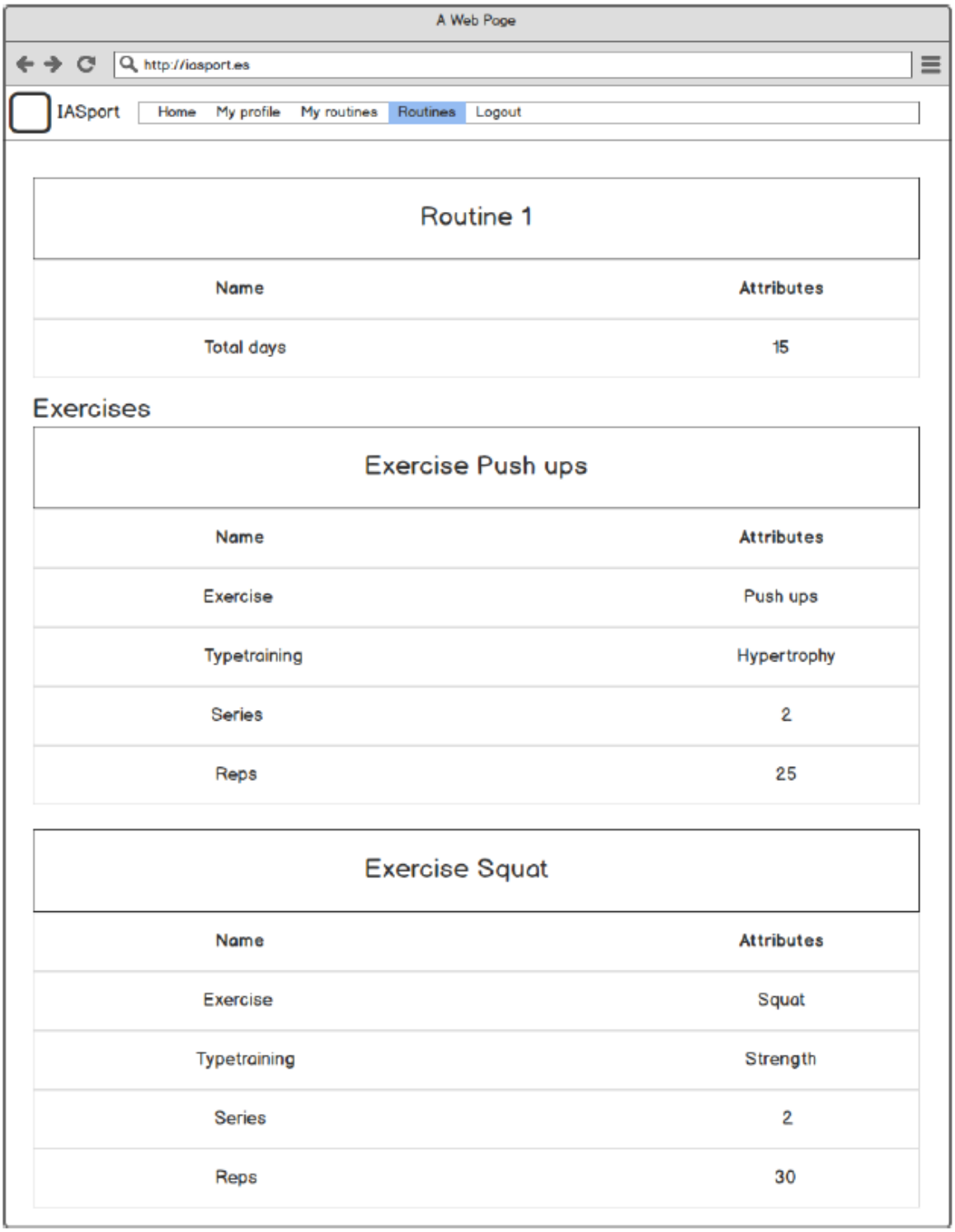


Ilustración 121 - Mockup web detalle rutina a la que no se está apuntado

14.4 Diagrama de casos de uso

Ilustración 122 - Diagrama de casos de uso