ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA

DESARROLLO DE UNA APLICACIÓN MÓVIL EN PG/CORDOVA PARA LA GESTIÓN DE DATOS, PARAMETRIZACIÓN Y EJECUCIÓN DE SERVICIOS WEB EN BIOINFORMÁTICA

DEVELOPMENT OF A MOBILE APP IN PG/CORDOVA FOR DATA MANAGEMENT, PARAMETRIZATION AND EXECUTION OF BIOINFORMATICS WEB SERVICES

Realizado por
Sergio Díaz del Pino
Tutorizado por
Oswaldo Trelles Salazar
Juan Falgueras Cano
Departamento
Arquitectura de Computadores

UNIVERSIDAD DE MÁLAGA
MÁLAGA, SEPTIEMBRE, 2016

Fecha defensa:
El Secretario del Tribunal

Resumen: A lo largo de este documento se describe el plan de trabajo para el desarrollo de una aplicación basada en web, especialmente diseñada para móviles pero accesible desde cualquier navegador, así como sus características y su funcionalidad. Empezando por las razones que motivan este trabajo, la definición de los objetivos del mismo y el estado del arte, se describen también el marco de desarrollo y las distintas etapas del desarrollo software. El resultado es mORCA, una aplicación basada en web especialmente diseñada para dispositivos móviles capaz de navegar a través de repositorios, descubrir y ejecutar servicios bioinformáticos y consultar resultados de los mismos. Además la aplicación ha sido portada a las principales plataformas móviles (Android e iOS) con el uso de marcos de trabajo como PhoneGap y Cordova, que mediante instalación permiten mejorar la experiencia del usuario final guardando información en el propio dispositivo.

Palabras claves: Bioinformática, Web, Movil, Phonegap, Cordova

Abstract: In this document we describe the road map for the development of a web-based application specially developed for mobile devices but also accesible for any browser. Its characteristics and its functionallity is also described. It starts for the reasons that motivates this work, the definition of the goals and the state of the art and continues with the description of the development framework and its differents stages. The result is mORCA, a web-based application specially developed for mobiles devices that allows the user to browse through repositories, discover and execute bioinformatics services and check the results. The application has been wrapped to work in the main mobile platforms (Android and iOS) using frameworks as Phonegap and Cordova, which through an installation improves the final user experience by storing some data in the device.

Keywords: Bioinformatics, Web, Mobile, Phonegap, Cordova

Tabla de contenidos

Capítulo 6. Desarrollo

6.1 Prototipos
6.1.1 iOS
6.1.2 Tecnologías Web + Cordova/PhoneGap
6.2 Tecnologías
6.2.1 jQuery
6.2.3 jQuery Mobil
6.4 Versión Web-App
6.5 Versión empaquetada

Capítulo 7. Ejecutando servicios en la nube con mORCA

7.1 Navegando con mORCA
7.1.1 Ejemplo 1
7.1.2 Ejemplo 2
7.1.3 Ejemplo 3

Capítulo 8. Discusión y conclusiones

Bibliografía

Apéndices

Anexo 1. Servicios web

1.1 Introducción
1.2 Tecnologías
2.2.1 SOAP
2.2.2 REST
1.3 Repositorios

Anexo 2. API Modular

2.1 Introducción
2.2 Módulos
2.3 Arquitectura
2.3.1 Interfaz
2.3.2 Acceso

Anexo 3. Estado del arte

Capítulo 1. Introducción

1.1 Motivación

El crecimiento del uso de dispositivos móviles en los últimos años [1] [2] y sus características implícitas tales como la ubicuidad de acceso a internet y su portabilidad, en combinación con su aumento de potencia, han hecho de estos dispositivos unos aliados imprescindibles no solo en nuestra vida personal sino también en el aspecto laboral [3] [4].

La bioinformática, definida como el uso de técnicas y herramientas computacionales para el análisis de la información biológica [5], tradicionalmente ha ofrecido servicios computacionales a través de la Web, y más recientemente basándose en el concepto de servicios webs [6] como por ejemplo aquellos de consulta de base de datos biológicas tales como GenBank [7] o UniProt [8], o servicios para la comparación y alineamiento de secuencias, tales como BLAST [9], FASTA [10], etc. Por estas razones no debe quedarse atrás en esta nueva tendencia de uso y explotación de las nuevas tecnologías disponibles.

Por ello, en este proyecto se propone la el diseño, implantación y pruebas de un cliente software capaz de descubrir, componer, invocar y monitorizar la ejecución de servicios Web. La demostración de su utilidad se realizará sobre servicios y herramientas bioinformáticas.

Para ello, centrándonos en el modelo cliente-servidor, apoyado en la disponibilidad de repositorios de servicios y de las tecnologías web para llevar a cabo un cliente ligero, con respuesta rápida y capaz de ser ejecutado en la mayoría de los dispositivos más importantes. Los repositorios de servicios equivalen a una base de datos, con interfaz de Servicio Web, que contiene la descripción del servicio, parámetros, forma de invocación, etc. Son la tendencia actual, como lo demuestra su utilización en los proyectos ELIXIR [11], la infraestructura computacional Europea para la bioinformática (equivalente al CERN en física)..

1.2 Objetivos

En los últimos años hemos experimentado un crecimiento inesperado de los dispositivos móviles, tanto es así que en todos los ámbitos de la informática se están desarrollando herramientas y aplicaciones especialmente diseñadas para ser usadas desde estos dispositivos. Nuestra opinión es que la bioinformática no debe quedarse atrás en esta nueva oportunidad.

Aunque ya existen algunas aplicaciones bioinformáticas en los mercados tradicionales de aplicaciones para dispositivos móviles como *Biocatalogue* (link), *SimAlign* (link), Oh *Blast it!* (link) que cumplen una función específica, desarrollar una de ellas para cada uno de los servicios que un repositorio puede llegar a ofertar seria poco razonable.

El objetivo de este proyecto es el diseño, desarrollo y pruebas de un cliente interactivo, aplicación web, con interfaces auto adaptativas a los distintos dispositivos, tanto multitáctiles móviles como equipos de sobremesa, en forma de aplicación híbrida, que proporcione una funcionalidad intuitiva y fácil de usar, para configurar y activar los servicios web. La aplicación además gestionará los repositorios de datos, permitiendo su referencia remota, la búsqueda y descubrimiento de servicios, y en particular diseñada para soportar la gestión de grandes ficheros, como su entrada local a través de la interfaz, sin que el tipo de dispositivo y su menor capacidad de transferencia puedan suponer un cuello de botella para el uso de los voluminosos ficheros que actualmente se manejan en bioinformática.

Dicho cliente se desarrollará con tecnologías de software libre y estándares abiertos. Particularmente HTML5, JavaScript, PhoneGap/Cordova, y haciendo uso de la MAPI de acceso [12] desarrollado por el grupo de investigación Bitlab (www.bitlab-es.com) que actuará como backend en el servidor.

1.3 Características del cliente

Los requerimientos básicos para el desarrollo del cliente se han establecido en los siguientes puntos:

- Consultar, navegar, buscar y descubrir servicios sobre repositorios.
- Generar interfaces dinámicas para los servicios en base a su descripción contenida en los repositorios
- Ejecutar servicios, monitorizarlos y consultar resultados
- Descarga y subida de ficheros hacia y desde la nube.
-

1.4 Métodos

Para el desarrollo del cliente seguiremos un proceso iterativo e incremental, dividiendo el proyecto en pequeñas tareas las cuales contarán con las clásicas fases de un proceso de este tipo: análisis, diseño, implementación y pruebas

Fases de trabajo

1. Toma de requisitos y casos de uso de la aplicación. [30 horas]

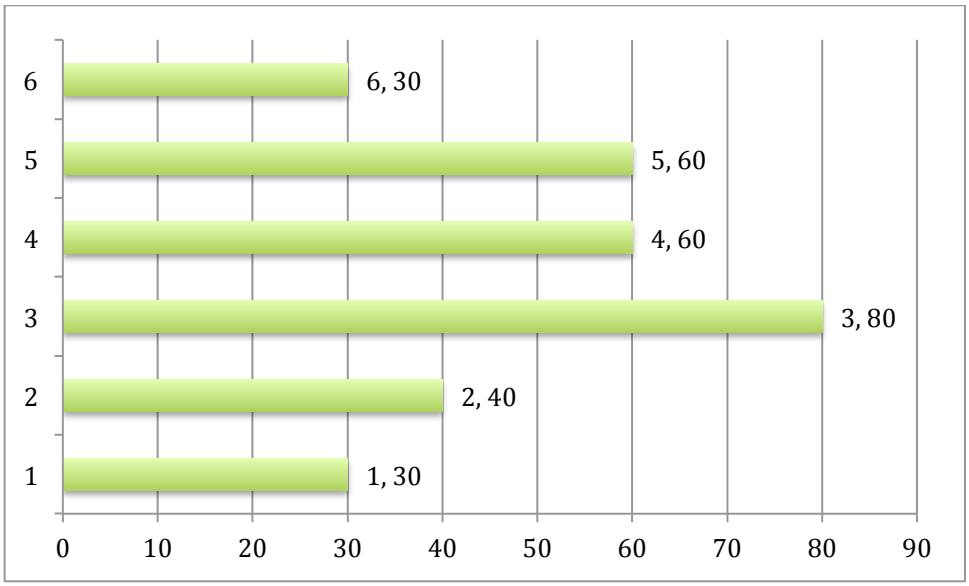
2. Prototipo de la aplicación: desarrollo de una aplicación que permita la consulta de los servicios y la navegación a través de ellos. Definición del aspecto de la aplicación [40 horas]

3. Interfaces dinámicas: Desarrollo de un algoritmo que permita generar interfaces dinámicas para cada uno de los servicios cuyas especificaciones se encuentren en el repositorio. [80 horas]

4. Ejecución de servicios: Implementación de la ejecución de servicios a través de la librería MAPI. [60 horas]

5. Gestión de ficheros: Implementación de la subida y descarga de ficheros y posterior implementación del uso de los mismos en las llamadas a servicios. [60 horas]

6. Pruebas y documentación [30 horas]



Capítulo 2. Metodología de desarrollo

2.1 Introducción

Para el desarrollo del proyecto empecé conociendo las herramientas que el grupo de investigación había desarrollado y empleaba para trabajar. Esto junto con la documentación de los mismos me ayudó a comprender mejor cuales eran las tareas del equipo y cómo funcionaba el sistema en el que basaban dicha actividad.

Una vez conocido los programas tuve acceso a los repositorios que contenían los códigos fuentes de dichos programas, los cuales me hicieron conocer a un nivel de profundidad mayor la arquitectura del sistema así como para reconocer muchos de los conceptos que había aprendido durante mi vida de estudiante.

2.2 Estado del arte

Era indispensable saber en qué punto se encontraba la bioinformática con respecto a nuestra idea de desarrollo, para ello se elaboró una lista de aplicaciones bioinformáticas disponible en las *Stores* oficiales de los sistemas mayoritarios (Ver Anexo 3) y se probó su funcionalidad.

Aún habiendo una cantidad de apps superior a la que en un principio nos esperábamos la funcionalidad de la mayoría de ella se limitaban a pequeñas aplicaciones con información biológica o aplicaciones capaz de ejecutar un servicio concreto. Muchas de ellas con interfaces inadecuadas o impracticables.

2.3 Desarrollo Ágil

Debido a las características del sistema, adoptamos un modelo de desarrollo ágil. Para ello nos basamos en Scrum, un modelo de desarrollo ágil que promueve un desarrollo incremental basado en diferentes fases de desarrollo que se solapan unas con otras.

Para ello definimos unos requisitos de la aplicación, seguidos de pequeños sprints que fueron acompañados de reuniones con los tutores del proyecto así como con otros miembros del equipo de investigación, completando con pequeños entregables las diferentes iteraciones.

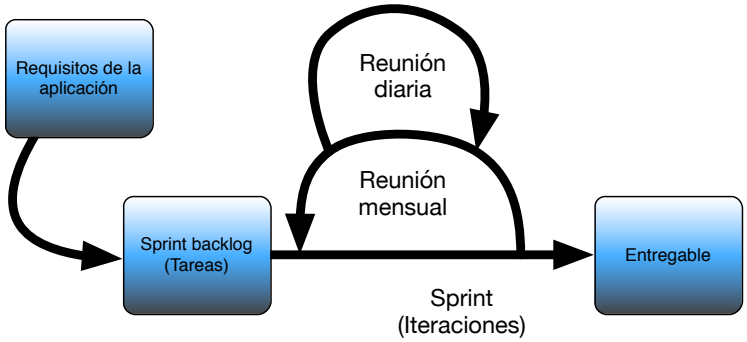


Figura 1. Diagrama del concepto de Scrum

2.4 Iteraciones

Podemos dividir el desarrollo del proyecto en 5 iteraciones.

La primera iteracción consistió en una toma de contacto con las tecnologías disponibles. Para ello creamos dos prototipos, uno desarrollado con tecnologías web y otro de forma nativa, en los que profundizaremos en la parte de implementación, así como un *technical review* que nos sirvió para discutir en reuniones cual de dichas tecnologías satisfacía mejor nuestras necesidades.

Una vez definidas las bases del proyecto, en la *segunda iteración*, realizamos una primera toma de requisitos relacionado con el Catálogo de servicios. Estos incluían la necesidad de trabajar con varios repositorios, comunicarse con MAPI para obtener los servicios, login de usuarios…

Ha sido una de las etapas más difíciles debido a que venía acompañada de la primera toma de contacto con el sistema de backend, las peticiones SOAP e incluso las primeras discusiones sobre cómo representar la información.

La *tercera iteración* venía marcada por una de las partes más importantes de la aplicación: la necesidad de tener que generar interfaces de forma dinámica de forma que fuese compatible con todos los servicios registrados.

Esto no solo supuso un reto a nivel tecnológico sino también a nivel de diseño. Algunos servicios poseen una cantidad considerable de parámetros por lo que la forma de mostrarlos y qué facilidades dábamos para que fuesen introducidos cobraban suma importancia.

Además, a nivel técnico nos obligaba a tener constancia de todos los tipos de parámetros, de sus valores permitidos, de los obligatorios y los opcionales…

La *cuarta iteración*, centrada en el módulo de ejecución de servicios, ha sido quizás la más rápida, debido a que, una vez representados e introducidos los parámetros era cuestión de organizarlos, empaquetarlos y mandarlos al servidor para que los procesase. Fue aquí donde aprovechamos las ventajas de AJAX para obtener los resultados generados al finalizar la ejecución.

La *última iteración*, que se centraba en la gestión de ficheros, puede ser dividida en dos partes. Aunque ambas compartían requisitos esenciales se desarrollaron dos sistemas de manejo de ficheros. Uno de ellos en el propio servidor y otro haciendo uso de las ventajas de Amazon S3.

Esto implicó primero discusiones sobre qué método de almacenamiento en la nube era más recomendable implementar y, además, la documentación y pruebas del mismo antes de ser usado en la aplicación.

Capítulo 3. Aplicación

3.1 Introducción

MORCA, nuestra aplicación móvil para la ejecución de servicios bioinformáticos en la nube, siguiendo los estándares del lenguaje WSDL y haciendo uso de los repositorios de servicios, es capaz de trabajar tanto con distintos servicios como con distintos repositorios de manera dinámica, ofreciendo una interfaz especialmente diseñada para dispositivos móviles. Funcionalidad que puede apreciarse de manera superficial en la Figura 2, donde se muestran los casos de uso.

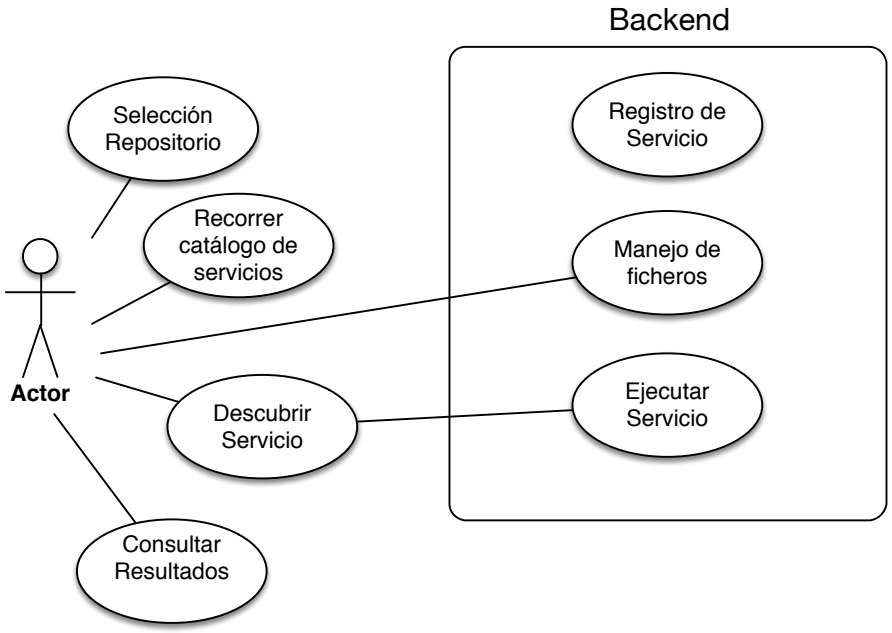


Figura 2. Casos de uso del sistema a desarrollar. El usuario final podrá navegar entre distintos repositorios y los servicios que los componen, ejecutarlos y consultar los resultados.

3.2 Partes del Sistema

Desde el comienzo del desarrollo de MORCA tuvimos claro que necesitaría no sólo de la parte cliente si no de una parte de backend encargada de procesar los datos y ejecutar los servicios.

En nuestro caso, el grupo ya contaba con un backend funcionando, MAPI, del que hablaremos en el Anexo 1. Dicho backend nos permitía acelerar el proceso de desarrollo, centrándonos en la parte del cliente.

Como podremos apreciar más adelante mORCA y MAPI, aunque independientes, gozarán de total comunicación entre ellas, situación clásica del modelo Cliente-Servidor.

3.3 Requisitos

A la hora de desarrollar mORCA nos encontramos con una serie de requisitos indispensables que nuestra aplicación debería cumplir.

- *Catálogo de servicios:*

  Debía ser compatible con los repositorios actuales y permitir explorar y buscar a través de ellos. El número de servicios en bioinformática no es sólo muy grande, sino que está en continuo crecimiento, es por ello que las herramientas para agruparlos y catalogarlos se hacen indispensables.

- *Composición de parámetros de servicios:*

  La generación de interfaces dinámicas conocidos los parámetros sería el punto fuerte, permitiendo trabajar con infinidad de servicios con interfaces adaptadas sin la necesidad de desarrollar cada una de forma individual. Esto será posible gracias a los metadatos con los que se registran los servicios.

- *Invocación:*

  Una vez encontrado el servicio y teniendo la interfaz generada y adaptada, los servicios debían poder ser invocados desde el propio teléfono y permitir la consulta de sus resultados.

- *Manejo de ficheros:*

  La mayoría de servicios necesitan de ficheros de datos, por ello era necesario implementar un sistema de ficheros en la nube que nos permitiera transferir archivos y hacer uso de ellos a la hora de ejecutar un servicio.

3.4 Requisitos de la interfaz

Uno de los principales problemas que se nos presentaba era cómo mostrar la información. Aunque hay varios estudios tratan la transición correcta entre interfaces WIMP (Windows, Icons, Menus and Pointer) y las móviles (o táctiles) [13] el caso particular de las aplicaciones en biomedicina y bioinformática requieren algunas consideraciones especiales debido al hecho de que la complejidad de los parámetros es muy elevada y requiere de soluciones gráficas a medida para evitar la saturación, especialmente en visualizaciones.

En la actualidad hay tres modelos de programación en lo referente a aplicaciones móviles [14], podemos hablar de:

- *Aplicaciones nativas:*

  Son desarrolladas en su totalidad en el lenguaje nativo del dispositivo. Esto implica varios desarrollos paralelos para cada una de las plataformas (iOS, Android, Windows Phone...) pero permite el acceso a todas las características del teléfono tales como GPS, acelerómetros…

- *Aplicaciones basadas en web:*

  Son desarrolladas con tecnologías web, lo que las hace totalmente independiente la plataforma y sólo requieren de un navegador web, a cambio el acceso a ciertas características del dispositivo está más restringido.

- *Aplicaciones híbridas:*

  Son una mezcla de las dos anteriores, se desarrollan usando tecnologías web pero se disfrazan de nativas haciendo uso de APIS capaces de interactuar directamente con el teléfono.

3.5 Ciclo de vida

El modelo de ciclo de vida clásico para navegar, descubrir y ejecutar servicios en la nube desde un cliente externo está ilustrado en la Figura 3. El proceso, generalmente, consiste en una serie de pasos que empiezan por registrar el servicio en el repositorio usando herramientas destinadas a tal tarea como Flipper [15].

Una vez que el servicio ha sido registrado sus metadatos son accesibles para el cliente. De esta forma el servicio puede ser seleccionado y este proporcionará los datos de su interfaz que será procesada por el cliente y mostrada al usuario. Además podemos hablar de la subida y bajada de ficheros a la nube para hacer uso de esos datos en la propia ejecución.

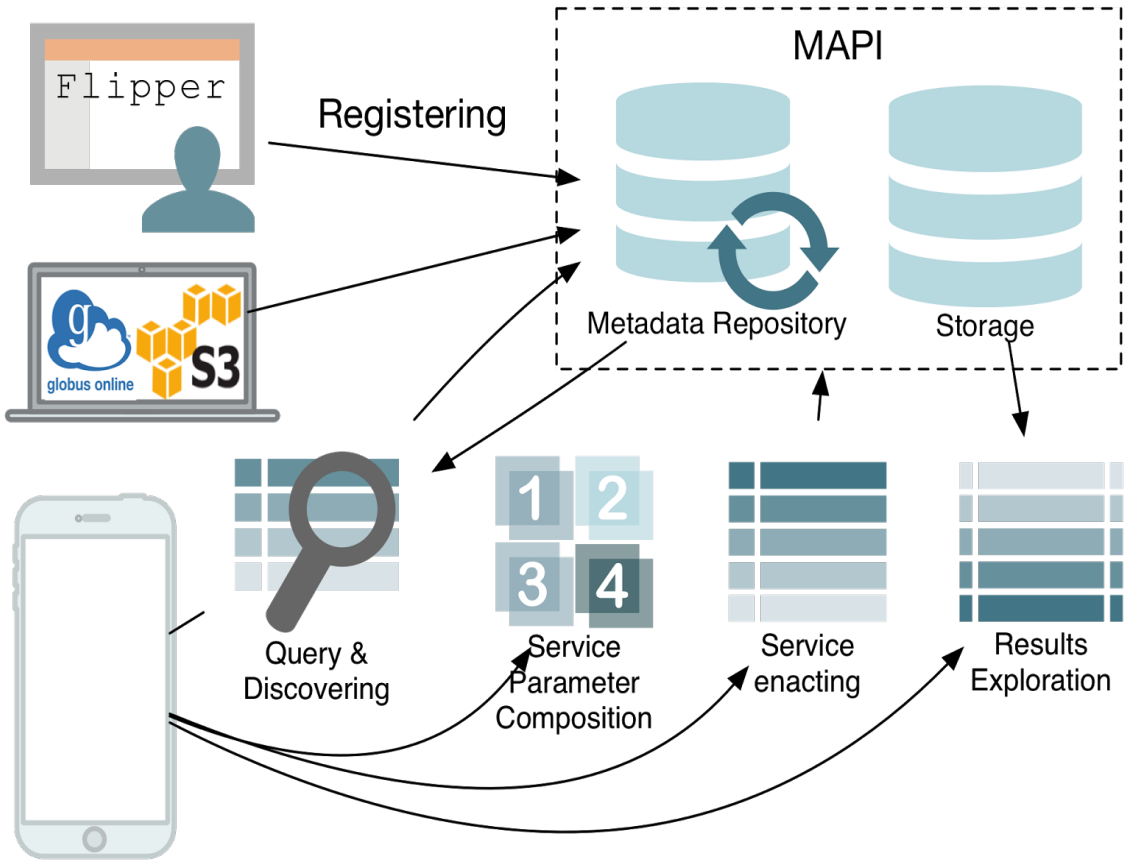


Figura 3. Ciclo de vida para el registro, descubrimiento, composición de parámetros, subida y bajada de ficheros y ejecución de los servicios .

3.6 Diseño de la Arquitectura

mORCA ha sido desarrollado usando MAPI (Modular API) como principal backend del sistema, permitiéndonos homogenizar el acceso tanto a repositorios como a Servicios Web.

MAPI se encargará de extraer la información de los repositorios, así como los metadatos de los servicios, traduciéndolos a un lenguaje accesible para nuestra aplicación.

En la parte del cliente la funcionalidad viene dada por la implementación del WSDL, que hemos dividido en pequeños módulos coincidentes con los requisitos que exponíamos en el punto 3.3.

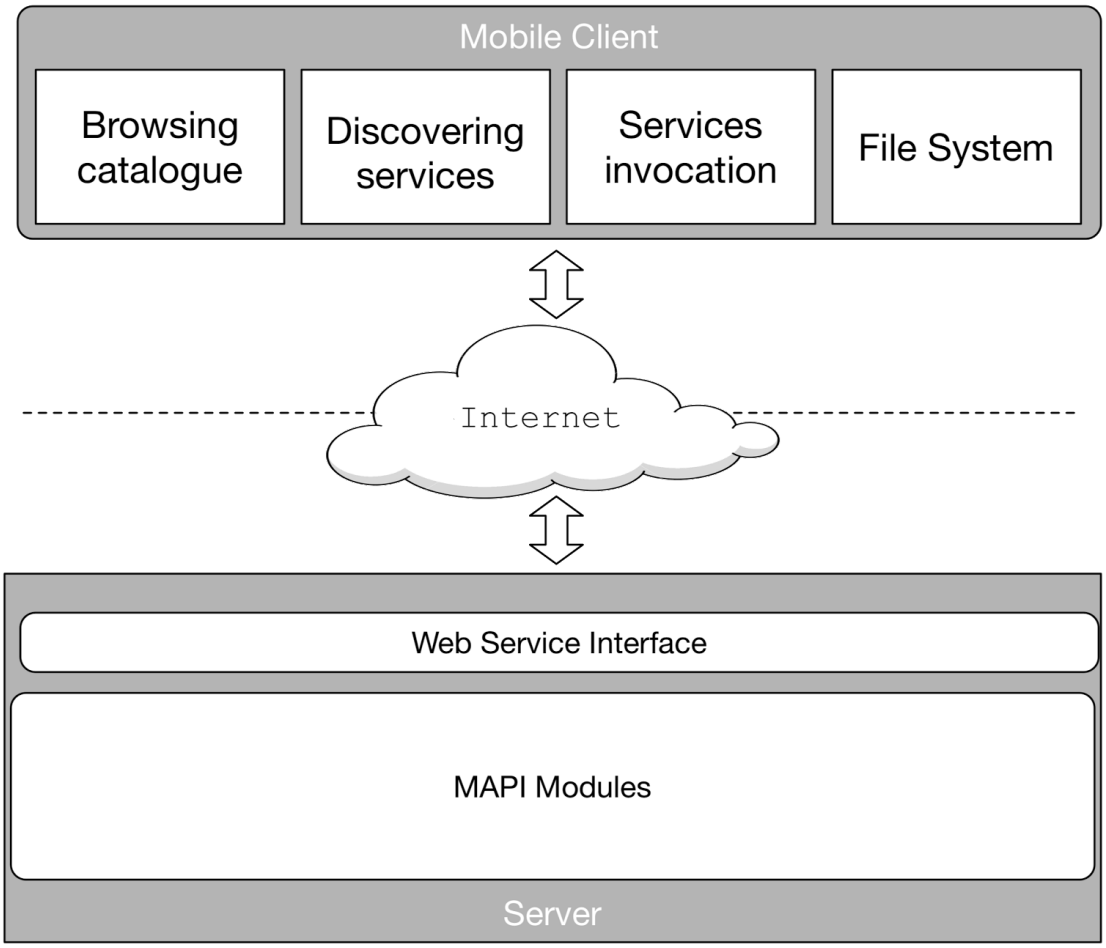


Figura 4. Arquitectura del sistema. Dividida en dos grandes grupos, Front-end y Back-end, compuestos de los diferentes módulos encargados del funcionamiento del sistema.

En la parte más baja, referente al lado del servidor, nos encontramos con MAPI con una capa superior de Servicio Web que actuará de comunicador entre ella y mORCA.

Ya en el lado del cliente tenemos los siguientes módulos:

- *Exploración de catálogos:*
    - En él se concentrarán los métodos necesarios para la exploración y búsqueda de servicios, permitiéndonos trabajar con varios repositorios.

- *Descubrimiento de servicios*:
    - En él estarán definidos los algoritmos encargados de procesar los parámetros y generar las interfaces

- *Invocación de servicios:*
    - Donde procesaremos los datos, la ejecución de los servicios y manejaremos los resultados.

- *Sistema de ficheros:*
    - Se encargará del manejo de ficheros en la nube.

Capítulo 4. Módulos de mORCA

4.1 Introducción

En este capítulo vamos a definir y comentar los módulos que componen nuestra aplicación. Además mostraremos una pequeña lista de los requisitos que definimos para cada módulo en las primeras etapas del proyecto.

4.2 Módulo de Exploración de servicios

El módulo de exploración de servicios de mORCA será el encargado de trabajar con los repositorios y sus listas de servicio. Deberá proveer una herramienta de búsqueda y capacidad de trabajar con varios repositorios.

Por lo tanto deberá cumplir con los siguientes requisitos:

- Obtención de las listas de servicios
- Capacidad multirepositorio
- Búsqueda mediante filtro de servicios
- Generación de árboles para la muestra de servicios

4.3 Módulo de descubrimiento de servicios

El módulo de descubrimiento de servicios de mORCA será el encargado de comunicarse con el servidor para obtener los datos de los parámetros de servicios así como de generar las interfaces de manera dinámica en función a esos parámetros y adaptarlas al dispositivo desde el que se vayan a ejecutar.
Los requisitos de este módulo son:

- Obtención de parámetros de servicios
- Generación dinámica de interfaces
- Diseño responsivo
- Búsqueda de servicios

4.4 Módulo de Invocación de servicios

El módulo de invocación de servicios de mORCA será el encargado de procesar los parámetros de entrada, encapsularlos y mandarlos al servidor para que el servicio sea ejecutado. Además, deberá recoger los resultados que arroje la ejecución de dicho servicio.

Los requisitos de este módulo son:

- Procesar los parámetros de entrada
- Ejecución del servicio en el lado del servidor
- Recuperar los resultados tras la ejecución

4.5 Módulo de sistema de fichero

El módulo de Sistema de ficheros será el encargado de manejar los ficheros tanto en el dispositivo como en la nube.

Los requisitos de este módulo son:

- Subir y descargar ficheros del servidor
- Gestión de ficheros
- Compatibilidad con Amazon S3

Capítulo 5. Interfaces móviles en mORCA

5.1 Introducción

La transición entre una aplicación WIMP para un entorno de escritorio a una interfaz móvil para un entorno táctil implica un cambio completo en la forma de interacción con el usuario. Las limitaciones en el tamaño de la pantalla, la entrada de datos, la visualización de resultados… plantean una completa revaluación de las mismas.

Todas estas limitaciones deben ser tenidas en cuenta debido a la importancia en la relación entre el usuario y el dispositivo ya que un mal diseño puede hacer de una aplicación útil algo inusable.

Es por ello que cada uno de los módulos de mORCA han sido diseñados prestando especial atención a todas ellas, haciendo que el usuario se encuentre con interfaces familiares en el entorno móvil pero que aún no han llegado totalmente al mundo bioinformático.

5.2 Exploración del repositorio

Una visualización tipo TableView ha sido usada para representar el árbol de servicios como una lista navegable de carpetas y servicios. Esto provee al usuario de un conocimiento rápido de las principales categorías del árbol. Cada celda, con el nombre del servicio acompañado por un icono y una pequeña descripción, es navegable hacía el siguiente nivel del árbol o hacía la interfaz del servicio para ser invocado.

La fluidez de la vista se consigue guardando de forma local la información referente al repositorio, es decir, cacheando la lista de servicios, comunicándose con el servidor sólo para recibir los datos de un servicio concreto, junto con animaciones CSS3 que hacen a los paneles hijos aparecer de derecha a izquierda.

5.3 Descubrimiento e invocación de servicios

En los módulos para el descubrimiento e invocación de servicios encontramos las interfaces de servicios generadas dinámicamente. Nuestra aplicación, después de que el usuario haya seleccionado un servicio de la lista del repositorio, manda una petición de consulta al backend. Este nos devuelve una lista de parámetros así como algunas datos referentes a ellos como pueden ser el tipo, los valores permitidos.. que se requieren para ejecutar el servicio. mORCA es capaz de construir la interfaz de usuario adecuada para solicitarlos con esta información. Este método no es siempre

el óptimo para construir una interfaz de usuario adecuada, de manera que se aplican heurísticas que permiten organizar en diálogos coherentes las interfaces, poner valores por defecto e incluso, permitir la descripción ad-hoc de la interfaz añadiendo esta descripción como metainformación al servicio.

Además, estas interfaces han sido diseñadas para generarse usando elementos HTML5 que junto con jQuery Mobile se adaptan de forma responsiva al tamaño de pantalla y facilitan la entrada de datos haciendo uso de botones grandes o 'selects' y 'checkboxes' adaptados a los teclados móviles.

5.4 Sistema de ficheros

El manejo de ficheros en los dispositivos móviles siempre supone una limitación debido al poco espacio con el que cuentan.

En este caso hemos optado por mantener en la nube un sistema de ficheros para cada usuario el cual permite tanto subir ficheros desde los dispositivos móviles que lo permitan o desde un ordenador como generarlos directamente desde la aplicación. Esto es buscando una secuencia mediante uno de los servicios o copiándola directamente desde nuestro terminal.

Las razones son varias:

a) Tamaño de los ficheros: Las capacidades de los teléfonos móviles oscilan, en general, entre los 16Gb y los 128Gb, manteniendose la mayoría los niveles más bajos. Es por esto que manejar ficheros de gran tamaño de manera local se hace, en muchos casos, imposible.

b) Relación tiempo de translado/coste: Enviar y recibir ficheros de manera local puede suponer un problema de coste debido al tamaño de los mismos.

c) Mayor accesibilidad: La mayoría de las veces estos datos se encuentran ya en repositorios estándares conocidos y no son resultado de experimentos locales.

Además nuestro sistema se ha integrado con Amazon S3 para importar ficheros.
Amazon S3 es un sistema de almacenamiento online ofertado por Amazon Web Services. Provee de almacenamiento a través de servicios web que funcionan mediante REST, SOAP o incluso BitTorrent

El diseño del módulo encargado de gestionar los ficheros permite hacer que cualquier otro sistema de almacenamiento online sea compatible desarrollando un pequeño plugin. Esto hace que protocolos tan conocidos como FTP, http o APIs de servicios populares como Dropbox, Mega... puedan tener su adaptación a mORCA.

Para ello se han utilizado representaciones clásicas de listas de ficheros, en este caso adaptadas a pantallas pequeñas, a las cuales acompañan iconos que permiten interactuar con los propios ficheros (Información, borrar, editar...)

Capítulo 6 – Desarrollo e implementación

6.1 Introducción

En este capítulo vamos a describir las diferentes tecnologías usadas durante el desarrollo del proyecto así como los caminos que nos llevaron a tomar dichas decisiones.

Además explicaremos las diferentes versiones finales en las que ha concluido la aplicación y los procesos para llegar a ellas.

6.2 Prototipos

A la hora de tomar la decisión sobre qué modelo de desarrollo elegir decidimos desarrollar un prototipo en lenguaje nativo para conocer mejor la plataforma y así juzgar desde la experiencia.

6.2.1 iOS

Se tomó la plataforma iOS como base de desarrollo, usando Objective-C como lenguaje nativo, y desarrollamos un prototipo que cumplia la función de comunicarse con MAPI de una manera rudimentaria para ver las posibilidades y el rendimiento.

Figura 5. Primer prototipo en iOS mostrando el arból de servicios.

El resultado de nuestra experiencia fue que iOS, aun comportándose de manera eficiente, está demasiado vinculado a Apple y suponía un problema para su futuro mantenimiento.

6.2.2 Tecnologías Web + Cordova/PhoneGap

Para nuestro prototipo usando tecnologías web desarrollamos una pequeña vista que obtenía un árbol de servicios desde MAPI y lo encapsulamos mediante PhoneGap. La ventaja primordial de usar PG/Cordova es la de hacer la aplicación una sola vez para todas las plataformas, usar un lenguaje de desarrollo común, como es JavaScript+HTML5 y, a la vez, poder instalar en cada plataforma el código suficiente como para que todo no dependa en todo momento de una conexión internet.

La filosofía de la web, está soportada por todas las plataformas y es razonablemente compatible entre ellas, además dispone de una amplia cantidad de frameworks para desarrollo móvil en la que todas están asimismo volcadas en la actualidad.

Cordova/PhoneGap [16] nos permite encapsular una única aplicación, hecha una sola vez para todas las plataformas, permitiendo así usar espacio físico del teléfono para cachear y mejorar la velocidad de la misma.

6.3 Tecnologías

La web está en continuo crecimiento así como también lo están sus marcos de trabajo. En la actualidad hay varias opciones a la hora de desarrollar aplicaciones web pero la mayoría de ella pasa por JavaScript.

Tanto es así que jQuery, un framework basado en JavaScript que facilita el acceso al DOM, se ha convertido en un indispensable de esta generación. Además, con el crecimiento móvil no tardó en salir un producto que combinaba a este último con un framework de interfaces móviles responsivas denominado jQuery Mobile.

6.3.1 jQuery

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos

HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Es libre y de código abierto, bajo licencias MIT y GNU v2, lo cual permite su uso tanto en proyectos libres y privados. Su principal característica es que ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código.

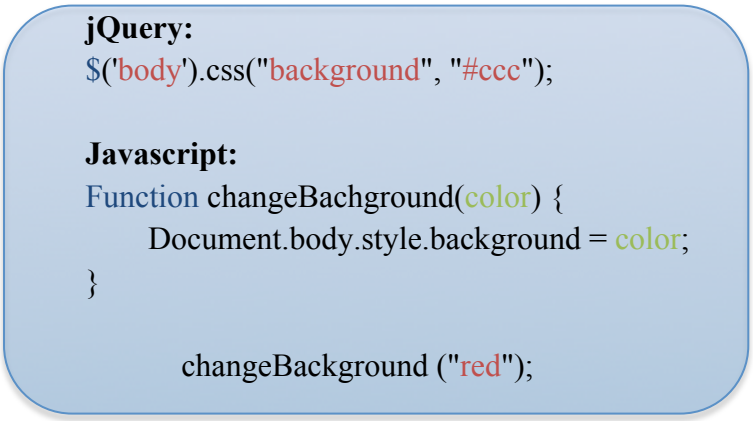


Figura 6. Diferencias de código para una pequeña función usando Javascript y jQuery.

6.3.2 jQuery Mobile

JQuery Mobile es un Framework optimizado para dispositivos táctiles que está siendo desarrollado actualmente por el equipo de proyectos de jQuery. El desarrollo se centra en la creación de un Framework compatible con la gran variedad de Smartphone y tablets, algo que se antoja necesario en los tiempos del acceso a internet móvil.

El Framework de jQuery Mobile es compatible con otros frameworks móviles y plataformas como PhoneGap y Worklight, lo que lo convierte en una potente herramienta de desarrollo.

```
success: function (soapResponse) {

      if (window.DOMParser) {
           parser=new DOMParser();
           xmlDoc=parser.parseFromString(soapResponse.toString(),"text/xml");
    } else  // Internet Explorer {
          xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
          xmlDoc.async=false;
          xmlDoc.loadXML(soapResponse.toString());
    }

          var token =
          xmlDoc.getElementsByTagName("loginWSReturn")[0].childNodes[0].nodeVa
          lue;

          //Create a cookie for the token
          var now = new Date();
          var time = now.getTime();
          time += 3600 * 1000;
          now.setTime(time);
          document.cookie = 'token=' + token + '; expires=' + now.toUTCString();
          document.cookie = 'username=' + user +'; expires=' + now.toUTCString();

          $.mobile.loading("hide");
},
```

Figura 7. Ejemplo de Callback de una petición SOAP para obtener los parámetros del servicio combinando jQuery y Javascript.

```
<div data-role="page">
  <div data-role="header">
    <h1>Página de ejemplo</h1>
  </div>
  <div data-role="main" class="ui-content">
    <p>Página de ejemplo para jQuery mobile </p>
  </div>

  <div data-role="footer">
      <h1>Texto del footer</h1>
    </div>
  </div>
```

Figura 8. Ejemplo de etiquetas de jQuery mobile. Los “data-role” le dicen que función cumplen los contenedores.

6.4 Versión web-app

La primera de las versiones de mORCA consiste en una versión totalmente web, accesible desde cualquier dispositivo sea móvil o no, y que permite su uso como web App.

De esta forma no es necesaria ninguna descarga en el dispositivo y tan sólo hace falta de un navegador web compatible.

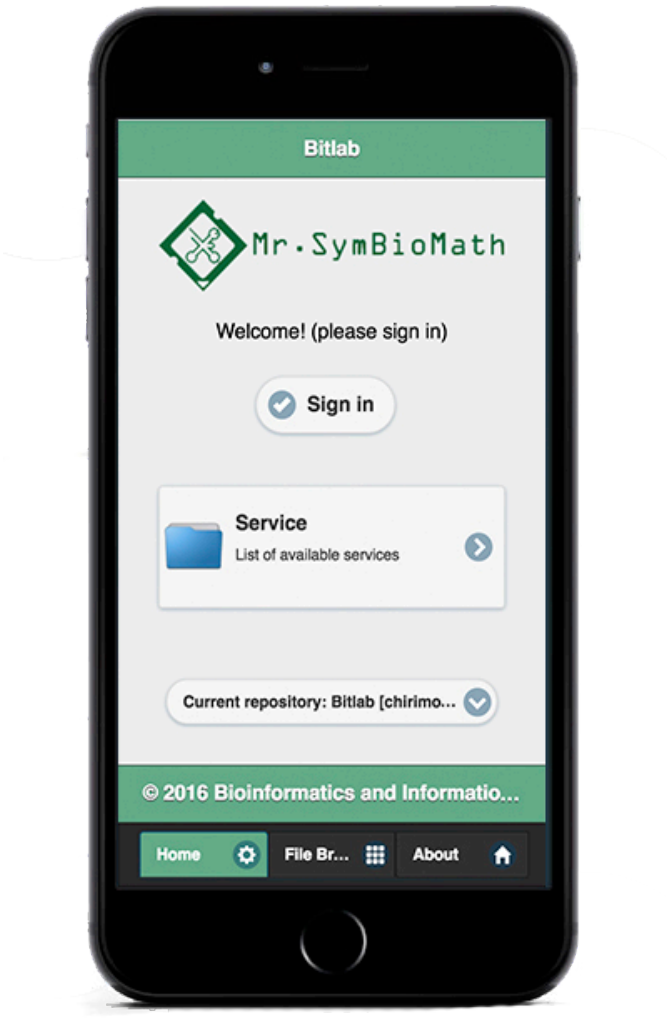


Figura 9. Acceso a la aplicación desde un dispositivo móvil.

6.5 Versión empaquetada

Como decíamos en el punto 4.4, hemos hecho uso de Cordova/PhoneGap para empaquetar nuestra aplicación en un .apk compatible con los dispositivos Android.

Será necesario descargar cada uno de los SDK's de los sistemas operativos a usar (iOS, Android, Windows Phone…) y empaquetar la app para cada uno de ellos. Esto nos permitirá mejorar no sólo en velocidad si no que nos permitirá acceder a los principales mercados de aplicaciones de las distintas plataformas (Google Market, Amazon Marketplace, App Store...)

Capítulo 7. Ejecución de servicios con mORCA

7.1 Navegando con mORCA

Con la intención de demostrar el potencial de proveer acceso a servicios bioinformáticos mediante dispositivos móviles procedemos a presentar tres ejercicios:

- El primero centrado en la búsqueda de un servicio y obtención de una secuencia.

- El segundo centrado en la ejecución de un servicio clásico (Blast).

- El tercero centrado en la ejecución de un workflow para el estudio de la homología de unas secuencias.

Los dos primeros ejemplos conforman en sí mismos un pequeño workflow: Obtendremos una secuencia usando uno de los servicios y más tarde, en el ejemplo dos, usaremos dicha secuencia para ejecutar un Blast.

En el tercer ejercicio el workflow será autómatico, ya que así estará definido en el repositorio y, además, incluirá el uso de servicios externos.

La complejidad de los ejemplos es mayor a medida que avanzan, mostrando en el primero una interfaz simple con poca carga de información, en el segundo una interfaz de un servicio más complejo, con una carga de parámetros mucho más densa y variada. El último lo centraremos en la obtención de los resultados finales para ser interpretados.

7.1.1 Ejemplo 1: Búsqueda de servicios

Este primer ejemplo incluye la búsqueda y el descubrimiento del servicio 'GetAminoAcidSequence'. Dicho servicio obtiene una secuencia de un aminoácido desde una base de datos biológica.

Para ello:

a) Será necesario identificarse en el sistema por motivos de seguridad.
b) Seleccionaremos el repositorio con el que queremos trabajar. En este caso será el repositorio por defecto 'Bitlab'.
c) Localizaremos el servicio en el catálogo ya sea mediante la navegación por el árbol o ayudándonos de la búsqueda superior.
d) Una vez el servicio ha sido seleccionado se generará la interfaz correspondiente mediante una rápida llamada para obtener los metadatos del mismo.
e) Con los parámetros rellenados correctamente procederemos a ejecutar el servicio.
f) Finalmente los resultados serán recibidos mediante AJAX y mostrados en la pantalla al usuario. A su vez un fichero se habrá guardado en nuestra carpeta de usuario.

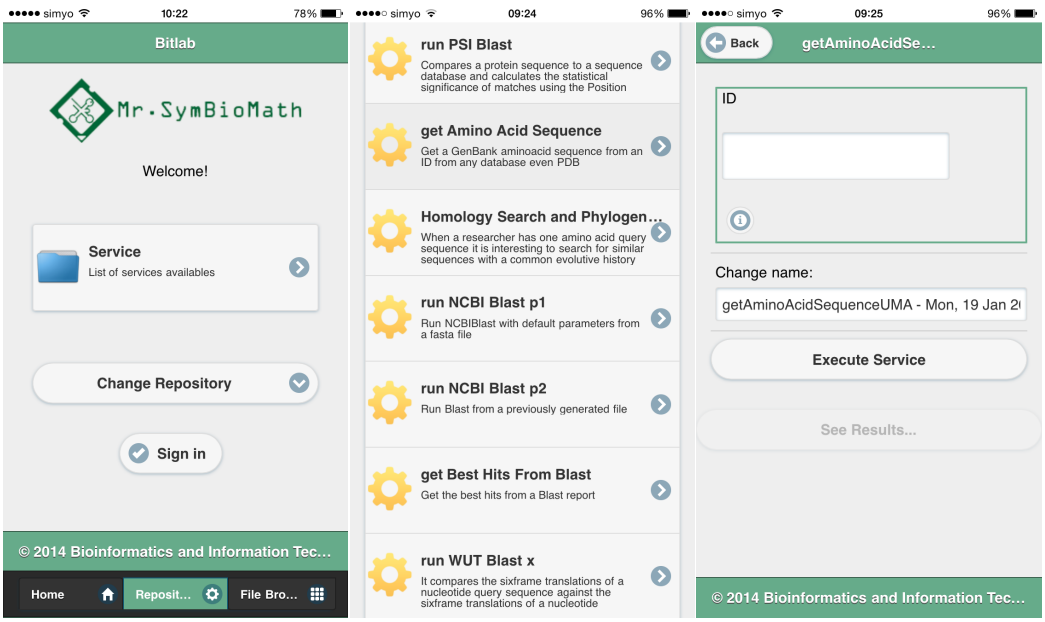


Figura 9. Interfaz para 'GetAminoAcidSequence' generada de forma dinámica.

7.2.2 Ejemplo 2: Ejecución de un servicio Blast

En este segundo ejemplo ejecutaremos el bien conocido Blast. Este servicio compara una secuencia dada contra una base de datos, en este caso SwissProt, y nos devolverá una lista de secuencias similares que encuentre en la misma base de datos.

Ya tenemos en nuestra carpeta de usuario la secuencia que hemos obtenido en el ejercicio anterior, aunque también tendríamos opciones como subir nosotros mismos el fichero o copiar y pegar la secuencia directamente.

Para ejecutar Blast el procedimiento a seguir es muy parecido al anterior:

a) Será necesario loguearse en el sistema por motivos de seguridad.
b) Seleccionaremos el repositorio con el que queremos trabajar. En este caso será el repositorio por defecto 'Bitlab'.
c) Localizaremos el servicio en el catálogo ya sea mediante la navegación por el árbol o ayudándonos de la búsqueda superior.
d) Una vez el servicio ha sido seleccionado se generará la interfaz correspondiente mediante una rápida llamada para obtener los metadatos del mismo.
e) A la hora de rellenar los parámetros usaremos el icono en forma de nube para seleccionar nuestro fichero generado.
f) Rellenaremos el resto de opciones y ejecutaremos el servicio
g) Los resultados serán mostrados por pantalla y guardados en nuestra carpeta de usuario.

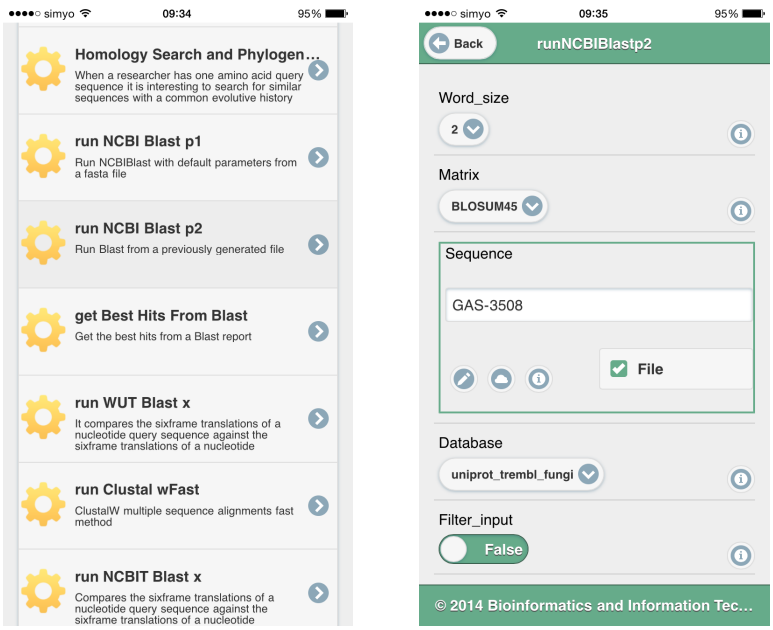


Figura 10. Interfaz para un servicio Blast generada de forma dinámica.

7.2.3 Ejemplo 3: Ejecución de un workflow

Para este último ejemplo vamos a ejecutar un pequeño workflow compuesto de varios servicios, incluyendo un servicio externo del EBI. Este workflow, registrado en el repositorio de Bitlab como 'Homology search and Phylogenetic study' usa un identificador de secuencia de entrada que se usará para descargar la secuencia de la base de datos de Unitprot [15] con la que ejecutaremos el resto de programas.

La lista de servicios a ejecutar es:

- GetAminoacidSequence: Descarga la secuencia de la base de datos (Unitprot)
- runEBIBlast. Ejecuta un BlastP (Búsqueda de homología usando una base de datos de proteinas) con la secuencia dada. La salida servirá de entrada para el siguiente servicio.
- getBestHitsFromBlast: Extrae los mejores hits del resultado del Blast usando un valor de corte (e-value) de 0.02. La salida es una colección de secuencias con sus respectivos identificadores.
- getAminoAcidSequenceCollection: Este servicio cogerá la lista de identificadores del servicio anterior y descargará sus respectivas secuencias.
- runClustalwFast: Realiza un alineamiento múltiple de secuencias usando el algorítmo ClustalW instalado en el EBI.
- runCreateTreeFromClustalw: Finalmente, las relaciones obtenidas con el alineamiento multiple se usan para construir un arbol filogenético con formato Newick. Este tipo de árbol es útil para estudiar las relaciones evolutivas entre las secuencias descargadas.

Para ejecutar dicho workflow los pasos a seguir son parecidos a los anteriores ejemplos:

a) Será necesario identificarse en el sistema por motivos de seguridad.

b) Seleccionaremos el repositorio con el que queremos trabajar. En este caso será el repositorio por defecto 'Bitlab'.

c) Localizaremos el servicio en el catálogo ya sea mediante la navegación por el árbol o ayudándonos de la búsqueda superior.

d) Una vez el servicio ha sido seleccionado se generará la interfaz correspondiente mediante una rápida llamada para obtener los metadatos del mismo.

e) Con los parámetros rellenados correctamente procederemos a ejecutar el servicio.
f) Finalmente los resultados serán recibidos mediante AJAX y mostrados en la pantalla al usuario. A su vez un fichero se habrá guardado en nuestra carpeta de usuario.

Figura 11. A la izquierda, la interfaz del workflow generada dinámicamente. Después, el resultado del mismo en la aplicación y como arbol filogenético.

Capítulo 8. Discusión y conclusiones

Tal y como decíamos en el capítulo introductorio, el crecimiento de los dispositivos móviles se ha incrementado de manera casi exponencial en los últimos años. No solamente por el entorno social y la interacción humana que facilitan si no debido al fuerte impacto que están empezando a tener en el ámbito científico. Debemos aprovechar sus principales ventajas, como la ubicuidad de acceso.

Nuestra contribución reside en la experiencia obtenida del desarrollo de aplicaciones y servicios bioinformáticos y biomédicos. Estos entornos de desarrollo son bien conocidos por funcionar a través de la web interaccionando con el llamado 'Big data'. Y es aquí donde los dispositivos móviles han sido dirigidos.

Aunque estos dispositivos parten de una posición de desventaja al perder el ratón como dispositivo de entrada y contar con una pantalla pequeña como dispositivo de salida, creemos que ofrecen una posibilidad de interacción muy potente con los gestos táctiles tales como pellizcar, hacer zoom, rotar... con sólo pequeños movimientos de dedos sobre la pantalla.

No sólo las pantallas o la interacción táctil con el usuario son motivo de controversia: aunque algunos sistemas operativos como Android o Windows Phone permiten el acceso a su sistema de ficheros de forma tradicional, estos dispositivos no suelen contar con una gran capacidad de almacenamiento, estando la media en unos 16/32Gb, por lo que trabajar con grandes archivos, tales como algunas secuencias o comparaciones genómicas, directamente desde el dispositivo se antoja complicado.

En resumen, este proyecto presenta una aplicación móvil capaz de navegar a través de repositorios, buscar y descubrir servicios, generar interfaces responsivas de forma dinámica basadas en metadatos de servicios bioinformáticos y ejecutarlos desde el propio cliente. Con esto propone un primer acercamiento del uso de dispositivos móviles en campos científicos que todavía no aprovechan su potencial, como la bioinformática.

Bibliografia

[1] Internet Trends, 2013. Available from: <http://www.kpcb.com/insights/2013-internet-trends> [7 October 2013]

[2] The 'Mobile Only' Internet Generation, 2010. Available from: <http://www.slideshare.net/OnDevice/the-mobile-only-internet-generation> [7 October 2013]

[3] Laura Pfeifer Vardoulakis, Amy Karlson, Dan Morris, Greg Smith, Justin Gatewood, and Desney Tan. Using mobile phones to present medical information to hospital patients. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI'12, pages 1411–1420, New York, NY, USA, 2012. ACM.

[4] Robin Deegan. Managing distractions in complex settings. In Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services, MobileHCI '13, pages 147–150, New York, NY, USA, 2013. ACM.

[5] Attwood, T.K. and Parry-Smith, D.J. (1999), "Introduction to bioinformatics", Addison Wesley Longman, Harlow, Essex CM20 2JE, UK.ISBN 0 582 327881.

[6] Ison, J., Rapacki, K., Ménager, H., Kalaš, M., Rydza, E., Chmura, P., ... & Booth, T. (2015). Tools and data services registry: a community effort to document bioinformatics resources. Nucleic acids research, gkv1116.

[7] Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Sayers EW (2009). GenBank. Nucleic Acids Res. 2009 Jan;37(Database issue):D26-31. Epub 2008 Oct 21.

[8] Uniprot, C. (2010). "Ongoing and future developments at the Universal Protein Resource". Nucleic Acids Research. 39 (Database issue): D214–D219. doi:10.1093/nar/gkq1020. PMC 3013648free to read. PMID 21051339.

[9] Pearson WR, Lipman DJ: Improved tools for biological sequence comparison. Proceedings of the National Academy of Sciences, 1988, 85(8):2444–2448

[10] Tao Tao (2011-08-24). "Single Letter Codes for Nucleotides". [NCBI Learning Center]. National Center for Biotechnology Information.

[11] https://www.elixir-europe.org

[12] Karlsson, J., & Trelles, O. (2013). MAPI: a software framework for distributed biomedical applications. J. Biomedical Semantics, 4, 4.

[13] Cheung, Victor; Heydekorn, Jens, Scott Stacey and Dachselt Raimund. Revisiting hovering: Inter- action guides for interactive surfaces. In Proceedings

of the 2012 ACM International Conference on Interactive Tabletops and Surfaces, ITS '12, pages 355–358, New York, NY, USA, 2012. ACM.

[14] Native, HTML5 or Hybrid, Understanding Your Mobile Application Development Options, 2013 http://s3.amazonaws.com/dfc-wiki/en/images/c/c2/Native_html5_hybrid.png [7 October 2013] [1

[15] Oscar Torreño Tirado and Oswaldo Trelles; "Easily registering bioinformatics services metadata"; ECCB'14, the 13th European Conference on Computational Biology; Methods and technologies for computational biology; Strasbourg, France (September 7 to 10, 2014)

[16] https://cordova.apache.org

[17] https://www.oasis-open.org

[18] http://www.w3c.es

Apéndices

Anexo 1. Servicios Web

1.1 Introducción

Un servicio web es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de datos como internet. Son tanto OASIS [17] como W3C [18], que los define 'como un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web', son los comités responsables de la arquitectura y reglamentación de dichos servicios.

Desde su aparición los servicios web han tenido mucha aceptación tanto fuera como dentro de la comunidad científica debido a las ventajas que aportan:

- Interoperabilidad entre distintos software de forma estándar.
- El usuario sólo necesita un cliente para comunicarse con ellos.
- Fomentan la estandarización y el uso de protocolos que faciliten el acceso a su contenido.
- Permiten el uso de grandes bases de datos de forma remota.
- Pueden hacer uso de la capacidad de cómputo de la nube y no de la máquina personal del usuario.

Los servicios web son definidos por su propio lenguaje descriptor (WSDL de sus siglas en ingles Web Services Description Language), que no deja de ser un formato XML que describe las operaciones, entradas y salidas, así como la localización del servicio en cuestión. De esta manera el acceso a esos documentos nos da la posibilidad de conocer e interactuar –incluso de forma automática- con el servicio.

Este tipo de lenguajes no sólo son prácticos desde el punto de vista de la conectividad si no que su flexibilidad nos permite que los cambios a lo largo del tiempo no afecten al global mientras se mantengan las interfaces.

1.2 Tecnologías

Los Servicios Web hacen uso, principalmente, del lenguaje XML de forma que tanto la descripción de los mismos como la transmisión de los datos se hacen mediante este lenguaje. Transmisión que se realiza usando el protocolo HTTP.

En la actualidad existen dos protocolos para el acceso y uso de Servicios Web: REST y SOAP, incompatibles entre sí y con filosofías distintas.

1.2.1 SOAP

SOAP (del inglés, Simple Object Access Protocol) [REF o URL] es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por Dave Winer en 1998 [REF] llamado XML-RPC. Fue creado por Microsoft, IBM y otros y en la actualidad se encuentra también bajo el auspicio de la W3C.

Básicamente, SOAP [REF o URL] es un mecanismo de mensajería en una dirección sin estado. Utiliza XML y un mensaje se compone de varias partes:

- Envelope: Es el elemento raíz dentro del cual irá el resto del mensaje, en él se define no sólo el mensaje sino cómo procesarlo.
- Header: Contendrá información sobre el mensaje. En el ejemplo 1 serán los campos *Nombre* y *Apellido*
- Body: Contendrá la carga de datos del mensaje. En el ejemplo 1 serán los campos *Dirección* y *Edad*

(Nota: debido a su utilización y aceptación generaliza de algunos términos en su definicion inicial anglosajona, se mantendrán así a lo largo de este documento)

```
<?xml version='1.0'?>
<soap:envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
    <soap:header>
        <name>Sergio</name>
        <lastname>Diaz</lastname>
    </soap:header>
    <soap:body>
        <address>Obispo Gonzalez</address>
        <age>25</age>
    </soap:body>
</soap:envelope>
```

Figura 1. Ejemplo de mensaje SOAP. En él se pueden observar los campos arriba mencionados así como la información que los mismos contienen.

1.2.2 REST

La Transferencia de Estado Representacional (Representational State Transfer) o REST es una técnica de arquitectura software para sistemas distribuidos como la World Wide Web. Su creador, Roy Fielding [12], originó el término en su tesis doctoral en el año 2000 y desde entonces ha pasado a ser ampliamente utilizado por la comunidad de desarrollo con la intención de evitar la complejidad del protocolo XML.

En la actualidad el término REST se usa para describir cualquier interfaz Web simple que usa XML y HTTP sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el descrito en el punto anterior.
REST se basa en unas claves fundamentales de diseño:

- Un protocolo cliente/servidor sin estado. Cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Esto hace que ni el cliente ni el servidor tengan que recordar ningún estado de las comunicaciones previas. Esto, en la práctica, ha evolucionado en técnicas como las cookies o el localStorage para mantener estados en una sesión
-
- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información así como HTTP define un conjunto de operaciones, cuyas más destacables son POST, GET, PUT y DELETE y que podría equipararse a las operaciones CRUD que se requieren para la persistencia de datos.

- Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable a través de su URI.

- El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación mediante HTML o XML. Como resultado tenemos un sistema navegable simplemente siguiendo enlaces, sin requerir el uso de registros u otra infraestructura adicional.

Un concepto importante en REST es la existencia de recursos (elementos de información) que pueden ser accedidos mediante un identificador global (URI). Para manipular estos recursos los componentes de la red (clientes y servidores) se comunican a través de una interfaz estándar (HTTP) e intercambian representaciones de estos recursos.

La petición puede ser tramitada por cualquier número de conectores (por ejemplo clientes, servidores, caché, túneles, etc.) pero cada uno lo hace sin ver más allá de su propia petición. Es así que una aplicación puede interactuar con un recurso

conociendo el identificador del recurso y la acción requerida, sin la necesidad de conocer si existen caché, proxys, cortafuegos, túneles o cualquier otra cosa entre ella y el servidor que guarda la información. El único requisito es que la aplicación sea capaz de comprender el formato de la información devuelta que, por lo general, es un documento HTML o XML.

En la tabla 1 se muestra un ejemplo de una petición REST que comprende los datos estándar que podríamos encontrar en alguna base de datos con información personal de un usuario: Nombre, Apellido, Dirección, Edad.

```
<usuario>
    <nombre>Sergio</nombre>
    <apellidos>Sergio del Pino</apellidos>
    <direccion>Obispo González</direccion>
    <edad>25</edad>
</usuario>
```

Figura 2. Ejemplo de objeto REST.

1.3 Repositorios

Como hemos visto anteriormente, para utilizar un Servicio Web, el cliente únicamente necesita conocer el WSDL donde se describa la interfaz del mismo. Sin embargo, debido a la gran cantidad de servicios existentes en la actualidad y la dispersión de los mismos, se han desarrollado repositorios en los que se almacenan los enlaces al código WSDL de los servicios, permitiendo a los desarrolladores registrar sus aplicaciones y a los clientes encontrarlas fácilmente.

Anexo 2. API Modular

2.1 Introducción

MAPI es un marco de trabajo (framework), desarrollado por el grupo de investigación Bitlab del Departamento de Arquitectura de Computadores de la Universidad de Málaga, para uniformizar el acceso a distintos repositorios de Servicios Web facilitando de esta forma el desarrollo de clientes que trabajen con éstos.

El framework ha sido programado utilizando el lenguaje Java5 por lo que es multiplataforma y utilizable tanto en aplicaciones de escritorio como en aplicaciones web (haciendo uso de jsp), applets o servicios web.

Su nombre proviene de la idea de realizar un framework lo más desacoplado posible de manera que una aplicación que utilice sólo ciertas características de MAPI no necesite importarla por completo, sino únicamente los módulos usados y sus dependencias.

MAPI se encuentra dividida en dos capas claramente diferencias, la interfaz o módulo que es uniforme independientemente del repositorio al que estemos accediendo y los accesos que son específicos para cada repositorio y cuya tarea es traducir las peticiones provenientes de la capa superior al repositorio utilizando los protocolos específicos del mismo. Adicionalmente existen unos accesos especiales que permiten añadir una cache a un acceso concreto, agilizando las peticiones y permitiendo su uso sin conexión a la red.

Además de la interfaz programática realizada en Java, MAPI posee una interfaz de Servicios Web que proporciona aproximadamente la misma funcionalidad con la ventaja de poder ser utilizada por cualquier aplicación, sin importar el lenguaje en el que se haya desarrollado o el entorno en el que se ejecute.

2.2 Módulos

Como hemos comentado anteriormente, la principal característica de MAPI es su modularidad. Actualmente en MAPI consta de 8 módulos principales aunque el diseño no está cerrado a la aparición de nuevos módulos si resultara necesario.

Podemos dividir MAPI en los siguientes módulos:

- **ModularAPICore:** Es el núcleo de MAPI, en él se definen e implementan las utilidades comunes a todos los módulos.

- **DataType y Tool** son los módulos encargados de manejar las entidades presentes en el modelado de Servicios Web.

- **Namespace y FunctionalCategory** estructuran y ordenan dichas entidades agrupándolas de distintas formas.

- **Data** proporciona al desarrollador de aplicaciones herramientas para tratar los datos y convertirlos de un formato a otro.

- **FileSystem** permite almacenar estos datos ofreciendo servicios de persistencia.

- **ToolLocation** maneja los endpoints o puntos de acceso a los Servicios Web permitiendo el uso de mirros.

- **User** permite la administración de usuarios, MAPI contempla el uso de Servicios Web seguros que requieran algún tipo de autenticación, por ello también permite el control de usuarios permitiendo su creación y administración desde este módulo.

Además de estos módulos, MAPI define variantes seguras de los mismos con ligeros cambios para el control de acceso a los recursos.

2.3 Arquitectura

Todos los módulos de MAPI se encuentran divididos en dos capas bien diferenciadas: interfaz y acceso.

La interfaz define el modelo de datos utilizado por el módulo y es independiente del repositorio utilizado en cada momento. Mientras que es responsabilidad de los accesos transformar los datos y protocolos del repositorio para hacerlos coincidir con la interfaz común.

Cada módulo dispone de varios accesos especializados en cada uno de los repositorios que pueden ser manejados por MAPI (por ejemplo, BioMoby, INB, ACGT, WSDL, etc.) además de estos accesos, existe uno especial que actúa como capa intermedia entre la interfaz y el acceso proporcionando servicios de caché agilizando el sistema de haciéndolo independiente del tráfico de la red.

La utilización de un acceso u otro viene definido por un fichero de configuración (URL del repositorio, Proxy, fichero de caché, etc.). Por lo tanto, cambiar de repositorio o acceso es tan sencillo como editar un fichero XML, no siendo necesario ningún cambio en el código ni compilar de nuevo la aplicación.

2.3.1 Interfaz

El diseño de MAPI concibe la interfaz como un modelo común a la mayoría de los repositorios de Servicios Web, por lo que debe contener además de los campos más utilizados en la definición de los Servicios Web métodos para buscar, listar y organizar la información contenida en cada repositorio.

MAPI dispone de métodos para realizar las operaciones básicas realizadas sobre cualquier base de datos o capa de persistencia: Crear, Obtener, Actualizar y Borrar (Create, Retrieve, Update y Delete en inglés) normalmente indicado con el acrónimo CRUD.

Los métodos de creación de objetos nos permitirán añadir nuevas entidades al repositorio, útil cuando queremos registrar un servicio nuevo, organizar de forma estructurada los servicios (organizándolos por categorías), crear tipos de datos nuevos para nuestro servicio, dar de alta a un usuario en el sistema, etc.

MAPI permite obtener el listado completo de entidades por tipo (obtener el listado completo de servicios en una llamada, por ejemplo), también dispone de métodos para obtener un recurso completo o explorar un árbol de categorías solicitando las entidades descendientes de una dada. Además los listados devueltos

por MAPI implementan una funcionalidad adicional que permite un filtrado dinámico y completamente flexible de los resultados.

Cada entidad ofrece los mecanismos necesarios para editar cualquiera de sus campos, útil cuando deseamos cambiar la definición de un servicio, el nombre de un parámetro, la definición de un tipo de datos.

Como es obvio MAPI también ofrece la posibilidad de eliminar cualquier recurso que creamos que ya no es necesario o quede obsoleto del repositorio, pudiendo de esta forma tener un repositorio limpio y ordenado en todo momento. En el fichero de configuración se podrá indicar si se desean borrar automáticamente todas las dependencias del elemento a eliminar o por el contrario que se prohíba el borrado.

Cuando se utilizan los módulos seguros, todas estas funciones son monitorizadas y sólo serán ejecutadas en el caso de que el usuario tenga los permisos necesarios para llevarla a cabo. La interfaz es idéntica en ambos casos a excepción de los métodos necesarios para entrar y salir del sistema y manejar las sesiones.

Existes dos interfaces para MAPI:

- **Programática**: es la más usada, ha sido desarrollada en Java y permite utilizar todos los módulos de MAPI y explotar toda su funcionalidad.

- **Web Services**: Permite usar MAPI desde otros lenguajes de programación a costa de perder cierta funcionalidad debido a las restricciones propias de los Servicios Web

2.3.2 Acceso

Los accesos son la parte más importante de MAPI, en ellos reside la capacidad de ésta para leer múltiples repositorios distintos. Se puede entender un acceso como un conversor de interfaces tanto a nivel de datos como a nivel de operaciones.

MAPI define una interfaz que deben cumplir todos los accesos, con los métodos necesarios para cubrir las necesidades de la interfaz programática y un modelo de datos uniforme a la par que flexible.

La implementación de cada acceso deberá cubrir la funcionalidad de cada uno de los métodos definidos haciendo uso de los servicios que le proporciona el repositorio concreto que mapea. Por ello, es posible que un método de un acceso requiera de dos o más llamadas a servicios del repositorio.

Para evitar los posibles retrasos causados por este problema o por caídas temporales de los repositorios, se ha implementado un tipo de accesos especiales capaces de almacenar la información del repositorio en una caché, recudiendo considerablemente los tiempos de arranque y uso del acceso y permitiendo su uso incluso sin conexión a internet.

Además de cubrir la funcionalidad básica definida por MAPI, los accesos deben traducir las estructuras de datos de los repositorios a un modelo entendible por la capa superior. Para ello se utilizan pares atributo valor cuyas claves serán definidas en la interfaz del acceso.

Ésta definición del modelo de datos permite a la implementación del acceso incluir datos adicionales que pueden ser ignorados por la interfaz de MAPI e incorporados a ella en un futuro, o dejar campos sin utilizar en caso que el repositorio no contenga esa información.

Anexo 3. Estado del arte

Category*	App name	Performs	Advantajes	Dissatvantajes	Link
2	**CloningBench**	Helps to do cloning experiments, in every detail based on tables for each kind of enzimes, etc.	Up to the point, it helps with the details for this kind of experiments. Free	Not updated since 2011	https://itunes.apple.com/es/app/cloningbench-by-invitrogen/id458617777?mt=8
2	**Promega**	View, run and annotate Promega protocols. Run timers directly from the relevant steps, and when complete, email the annotated protocols for printing, etc	Interesting tool for this lab users. Free	Specific	https://itunes.apple.com/es/app/promega/id307546949?mt=8
1	**Phylogram**	Visualization of phylogenetic tree	Phylogram allows visualisation of the phylogenetic trees resulting from bioinformatics sequence analyses	Just for iPhone, not intuitive, does not hanlde landscape, can't get to file selection. Low interactivity. Difficult to zoom-handle-intertact even with small phylogenetic trees.	https://itunes.apple.com/us/app/phylogram/id399814043?mt=8&ls=1
1	**Biocatalogue**	Browse the catalogue for services, service providers. REST endpoints and SOAP Operations	Informational app, handles landscape and portrait. Free	Is more like a web site. Information is minimal, no links. Unorganized, plain lists, non hierarchical nor contents organized. Quite difficult to browse or search through.	https://itunes.apple.com/us/app/biocatalogue/id450120348?mt=8&ls=1
2	**EdgeBio**	It allows access to all content from the EdgeBio blog, website and their online communities	Unifies all digital resources of EdgeBio blog. Free.Interacts via scial networks	It is more a link to videos and on line resources than a tool or app. Not access from all countries	https://itunes.apple.com/us/app/edgebio/id515313935?mt=8&ls=1
1	**SimAlign**	iOS touch application that interfaces with NCBI BLAST service to align a provided sequence or Accession.	Saves all queries allowing users to recall and resubmit searches. iOS and Android	It is not confortable to use, many dialogs for the only task it can do: NCBI blast. It depends on registered users. It is unable to handle files, you must copy-paste or type the entire sequence in a small textbox. It is based on registering and not free accounts.	https://itunes.apple.com/us/app/simalign/id432818873?mt=8

1	**SimGene**	Interfaces with Simbiot, Ensembl, NCBI, etc, to retrieve annotation info for over 30-species, based on gene and symbol search	iOS and Android	Same user interface and uncomfortable handling as SimAlign. It is based on registering and not free accounts.	https://itunes.apple.com/us/app/simgene/id427772349?mt=8
1	**Oh BLAST It!**	Executes BLAST	Simple. Free	Only Android. It doesn't have a real graphical user interface, but a kind of black screen where everything is displayed and entering as command lines	https://play.google.com/store/apps/details?id=com.bioinformaticsapp
1	**Gene Aligner**	This program uses the Needleman-Wunsch and Smith-Waterman algorithms for pairwise gene global and local alignment, respectively.	Simple. Free. The sequences can be entered in two different ways: string of characters or NCBI accession number. The program can also generate a dot plot for the alignment.	Only Android. It doesn't have a real graphical user interface, but a kind of black screen where everything is displayed and entering as command lines	https://play.google.com/store/apps/details?id=itd_gene.activities
2	**NEB Tools /New England Biolabs Inc**	Enzyme finder for restriction enzyme finding information on any NEB enzyme	Complete, well documented and organised. Free	Quite specific for this labs	https://itunes.apple.com/es/app/neb-tools/id350346827?mt=8
3	**RCSB PDB**	Official mobile app of the RCSB PDB (Protein data bank)	Amazing database, with molecular 3D rendering.	Depends on internet connection. Only querying	https://itunes.apple.com/us/app/rcsb-protein-data-bank/id529153183?ls=1&mt=8
2	**PCR Essentials**	Interact Remotely with your ProFlex™ PCR system. product information on Taq Polymerases, dNTP, Gels and stains, RT-PCR products, cDNA products and master mixes, etc.	Smart interface, high quality app. Free	Specific	https://itunes.apple.com/es/app/pcr-essentials/id606990640?mt=8
3	**DrugBankWP**	Chemical, pharmacological and pharmaceutical database	Complete	WindowsPhone only	http://www.windowsphone.com/es-es/store/app/drugbankwp/968f1dbe-0e67-4c2a-b733-16bfcfd4f51c
1	**DNAApp**	Open ab1 files	Open and visualise sequences from Dropbox (in iOS). New versions save also in fasta format	Simple. Not well adapted to common gestures in mobiles.	https://itunes.apple.com/us/app/dnaapp/id854944694?mt=8

4	**Evolutionary Biology**	Concept to teach evolutionary concepts	Simple	More as web site than appication. Simple. It covers only quite a few topics in Bioinformatics, it's very simple and take no advantage of the interactive capabilities of the device. Not free	https://itunes.apple.com/us/app/evolutionary-biology/id513464425?mt=8&ls=
4	**Gen Tutor**	iOS student app for students taking a standard College level course in Genetics	Extensive and complete based on levels of knowledge. Free	A tool for students, following a course	https://itunes.apple.com/es/app/gene-tutor/id534062019?mt=8
4	**Gene Link**	Quick short course in molecular genetics	High quality material. Wide range and well organized number of topics. Tools for researchers. Free	Only tools for handling one sequence, simple ops	https://itunes.apple.com/es/app/array-genetic-tools-from-gene/id399712676?mt=8
4	**Nature ENCODE**	It helps to navigate through the 30 papers published in Nature, Genome Research and Genome Biology as the ENCODE Project	Amazing presentation of the state of the art in the most important papers published	iPad only	https://itunes.apple.com/es/app/nature-encode/id553487333?mt=8
4	**MyGenome/Illumina, Inc**	Explore real human genome	Graphic, health implications, educational, quite complete. Good graphics, a lot of information on genetic deseases. Cheap.	iPad only	https://itunes.apple.com/es/app/mygenome/id516405838?mt=8
4	**Human Genome**	Obtain graphical info between diseases and genes, syndromes or traits	Graphical, pedagogic, free	iPad only	https://itunes.apple.com/es/app/human-genome/id576939342?mt=8
2,4	**MIQE qPCR**	Helps with technical support on particular qPCR experiments. Based on characteristics, it offers articles and depending on your problem, particular advise	Essentially useful. Kept updated, with active version supporting. Free	Quite specific, documentative.	https://itunes.apple.com/es/app/miqe-qpcr/id423650002?mt=8
2	**Real-time PCR**	Clearly organised set of documentative resource on experiment preparation and results interpretation on PCR	Attractive, easy to use, a lot of information. Update. Free	Specific, documentative	https://itunes.apple.com/es/app/real-time-pcr/id523157743?mt=8

3	**Talking Glossary of Genetics**	By National Human Genome Research Institute, more than 250 common genetic terms pronounced and explained in an easy-to-understand way by leading scientists and professionals at the National Human Genome Research Institute (NHGRI).	Great pictures, explanations, comprehensive and balanced explanations on every genetic topic. Free	Heavy, only educational	https://itunes.apple.com/es/app/talking-glossary-of-genetics/id428340581?mt=8
3	**Protocolpedia**	For lab researchers. Offline access to hundreds of complete protocols	Light, complete. Well organised and supported by real users. Includes videos. Free	Not updated since 2011	https://itunes.apple.com/es/app/protocolpedia/id396334248?mt=8
3	**BioTechLCD**	Access to bioinformatics and genomic information	Free	More than an app is an informative tool-a blog	http://www.androidpit.de/de/android/market/apps/app/com.globetech.biotechdaily/biotech-daily
3	**BioGene**	A search and information tool for biological search. A quick reference for any gene	Really documentative. Well organised. Free	Specific for genes	https://itunes.apple.com/app/biogene/id333180084?mt=8
3,1	**BioGenie**	Multitask system to be used by biologist, scientist researchers, professors and students	Help using those services	Specific	http://bio-genie.com/node?page=2
4	**Gene Lab**	iOS educative game on creating creatures based on their genes	It permits to experiment essentially with the Mendel laws and mutation concept	Simple concepts, only recreational	https://itunes.apple.com/es/app/gene-lab/id320762232?mt=8
4	**Gene Pool**	Game. iOS	Amusement	Simple concepts	https://itunes.apple.com/es/app/gene-pool/id330408920?mt=8

* Categories

1: Independent for directly dealing with specific bioinformatic problems

2: Depends on particular companies to access their software

3: Accesses and browses specific databases

4: Educational walkthrough and/or game to explore and learn bioinformatics