Bloques, protocolos y notificaciones

Uno de los conceptos tal vez menos explorado en las aplicaciones móviles es la de reporte de clima o pronósticos del estado del tiempo. En parte porque Apple ofrece una muy buena aplicación. También porque los reportes suelen ser de terceros y se les deben pagar para usar dicha información. Sin embargo, existen algunas alternativas gratuitas y en esta ocasión exploraremos una de ellas: Open Weather Map. Y lo haremos creando una aplicación que despliegue el estado del clima, así como el pronóstico para los siguientes días, de una ciudad dada. El sitio de Open Weather Map es https://openweathermap.org/, y te invitamos a que le eches un vistazo antes de iniciar el desarrollo de la solución de este proyecto.

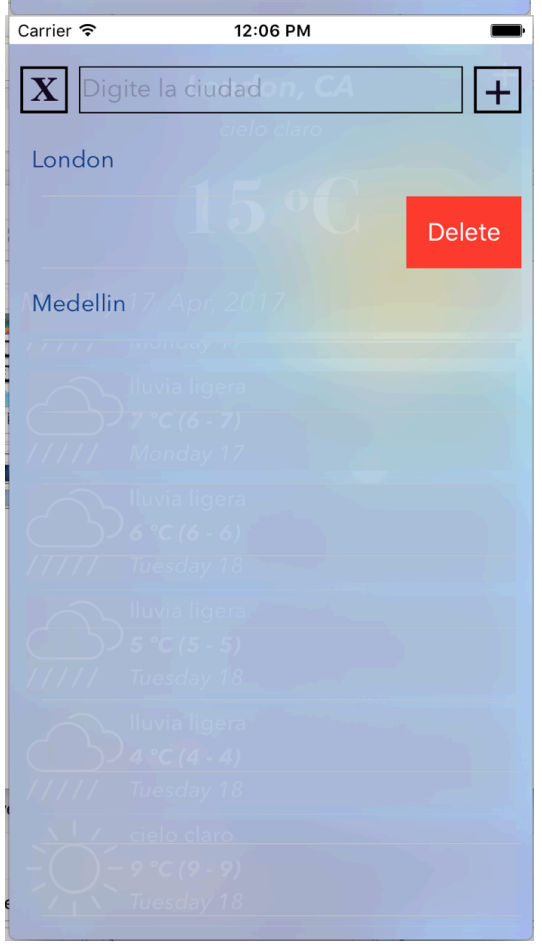
OBJETIVO:

Completar la implementación de una aplicación que despliega el estado del clima de un conjunto de ciudades dado, que inicia vacío y puede variar durante la ejecución. Implementar las funcionalidades y elementos faltantes, haciendo uso de programación orientada a protocolos y diferentes patrones de diseño como delegate, singleton y command. Por medio de las implementaciones descritas anteriormente, corregir y arreglar el proyecto actual para solucionar los errores presentes.

RESULTADOS:

En las siguientes imágenes se puede apreciar cuál será el resultado final esperado para la aplicación

Listado de hoteles en portrait y landscape:



1. La aplicación debe poder ejecutarse únicamente en iPhone en orientación vertical.
2. La solución debe implementarse sobre el proyecto base entregado: WeatherFeedBase.
3. La funcionalidad esperada de la aplicación es la siguiente: Cuando la aplicación inicia, deberá estar con la lista de ciudades vacía. La aplicación deberá desplegar la vista de administración de ciudades que es la que contiene una caja de texto, un botón de agregar y no de cerrar dicha vista, y una tabla que desplegará los nombres de las ciudades consultadas. Luego de que la caja de texto contenga algún texto ingresado por el usuario y de que el usuario active el botón de adicionar, se debe hacer una consulta al servicio del clima basado en la entrada del usuario. Cuando el servicio responda, se debe agregar un nuevo elemento a la lista de ciudades desplegando en este el nombre de la ciudad obtenida. La lista de ciudades deberá contar con la lógica y funcionalidad requerida para borrar ciudades ya consultadas. La manera de agregar ciudades será sólo mediante la entrada en caja de texto y la activación del botón adicionar. La manera de borrar ciudades será sólo mediante los elementos de edición de celda de la tabla.

3.1. Cada vez que esta vista sea cerrada, deberá informar a la vista de la colección con el fin de actualizar su contenido. Esta vista deberá desplegar la información de las ciudades agregadas mostrando el nombre de la ciudad, el estado actual del clima, la temperatura actual y la fecha de la lectura. Deberá desplegar también una lista con el pronóstico del clima para las siguientes horas y días conforme al resultado obtenido del servicio del clima. Todo esto en una celda de pantalla completa por ciudad, con un fondo cambiante que corresponda al estado actual del clima.

3.2. Ambas vistas, conforme a los pantallazos entregados anteriormente en este mismo documento.

4. El proyecto base cuenta con las vistas requeridas para el óptimo funcionamiento de la aplicación: La vista Weather View Controller que desplegará la lista de ciudades por medio de una colección, en donde cada celda será una ciudad y dicha celda ocupará toda la pantalla. Dicha celda contiene un encabezado en donde se desplegará el nombre de la ciudad, el estado y temperatura actual y la fecha de la lectura; debajo de dicho encabezado se tiene una tabla que desplegará la lista de pronósticos del clima para las siguientes horas y días de esa ciudad. La vista City Manager View Controller que desplegará una caja de texto en donde se digitará el nombre de la población, ciudad o región que se quiera consultar, junto con un botón de adicionar; cuando el botón de adición se active, se debe generar una consulta al servicio del clima preguntando por dicha locación y, en caso de recibir respuesta, la locación se agregará a la tabla que dicha vista contiene. No deben modificarse las vistas ni es necesario agregar más, pues conforme a los requerimientos y diseños de la misma, no se requieren de más elementos visuales.
5. El proyecto base cuenta con el grupo Utils en donde están definidos e implementados varios elementos del sistema. Uno de ellos es la clase Util en donde se definen funciones de uso global así como enumeraciones básicas del proyecto. También está definida la clase WeatherFeedServices que implementa todo lo relacionado a la conexión al servicio así como la respuesta a la consulta hecha. No se deben modificar ni alterar dichas clases ni sus funcionalidades, a excepción de la constante apiKey, definida en Util, en donde se deberá almacenar la API Key obtenida de OpenWeatherMap.org.

```
class Util {
    static let apiKey : String = "TU API KEY DE OPEN WEATHER AQUÍ"
```

6. Con respecto a la clase WeatherFeedServices y en donde corresponda, se debe implementar el bloque o closure respectivo, esperado por la función weatherData; dicho bloque será la implementación que obtendrá y manejará la data recibida del servicio y que equivale a la ciudad consultada con su respectivo pronóstico.
7. Otros elementos definidos en Utils son los dos protocolos que definen las estructuras de datos usadas en la aplicación: CityDataProtocol y ForecastDataProtocol. Se deben definir e implementar los elementos de dichos protocolos.

```swift
protocol CityDataProtocol {
    //TODO: Tu código aquí
}


protocol ForecastDataProtocol {
    //TODO: Tu código aquí
}
```

8. Un elemento adicional definido en Utils es el protocolo CitiesDataDelegate actualmente vacío. Se deben definir las funciones necesarias que se requerirán para que la clase helper de la tabla de ciudades se comunique con la clase controladora de la lista de ciudades. Dichas funcionalidades deberán ser implementadas en la clase CityManagerViewController que actualmente lo conforma.

```swift
class CityManagerViewController: UIViewController, UITextFieldDelegate, CitiesTableDelegate
```

9. El proyecto base cuenta con el grupo Models actualmente vacío. Se deben implementar las estructuras CityData y ForecastData dentro de dicho grupo y cada una de ellas deberá conformar los protocolos CityDataProtocol y ForecastDataProtocol respectivamente.

10. Dentro del mismo grupo Models, se deberá implementar una clase Singleton que centralice los datos obtenidos del reporte del clima de cada ciudad consultada durante la ejecución de la aplicación. Dicha clase singleton deberá definir la lógica y funcionalidad requerida para mantener los resultados de las ciudades consultadas y para proveer los datos tanto a la colección de ciudades, como a la lista de ciudades.

11. El proyecto base cuenta con el grupo Controllers en los que están definidos todos los controles que la aplicación requiere. Sin embargo, varios de estos controladores carecen de segmentos de código o funcionalidades que son necesarias para el correcto funcionamiento de la aplicación. Se deben implementar dichos segmentos de código, completando los espacios faltantes con el fin de que la aplicación compile y sea completamente funcional.

```swift
        self.hideLoading(animated: false)

        //TODO: Tu código aquí
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }

    @IBAction func tapUpAdd(sender: UIButton) {
        self.cityText.resignFirstResponder()

        //TODO: Tu código aquí
    }

    @IBAction func tapUpClose(sender: UIButton) {
        self.dismiss(animated: true) {
            //TODO: Tu código aquí
        }
    }
```

BUENA PRÁCTICA:

Procura tener organizado tu código, creando cada elemento en donde corresponde. Procura hacer uso de estructuras en vez de clases en donde sea posible.

Para este proyecto solo debes crear los elementos de código solicitados así como la implementación de la lógica en los lugares en donde sea necesario y/o se encuentre la etiqueta TODO: Tu código aquí.

Recuerda usar nomenclatura camelCase usando nombres y términos en inglés.

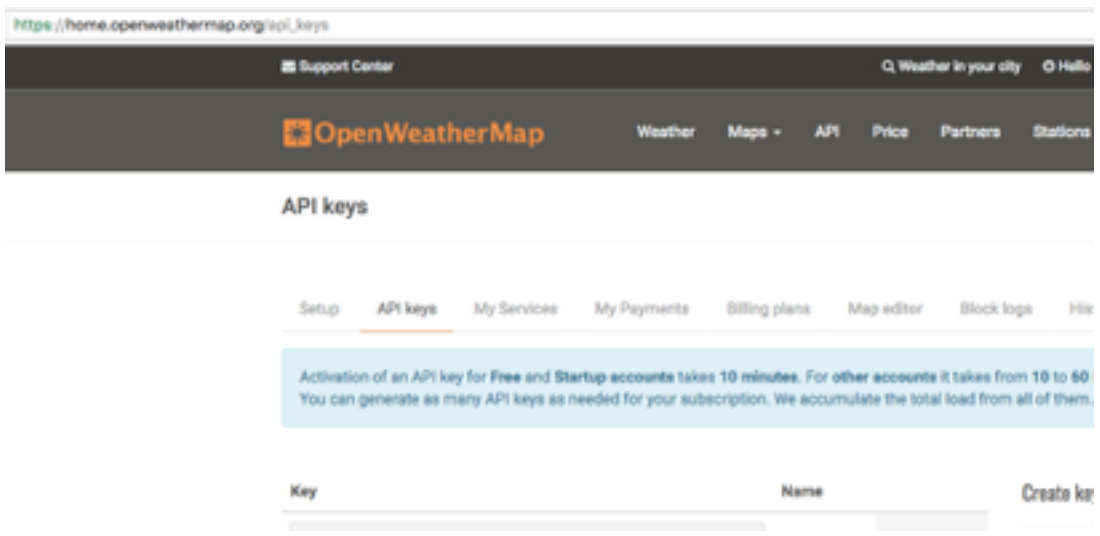
TEN EN CUENTA:

En la clase Util están definidas dos funciones que te serán muy útiles a la hora de formatear u obtener ciertos objetos relacionados a la información a desplegar. La función iconWeather te ayudará a obtener tanto el ícono como el fondo que corresponda al estado actual del clima con base en el código del mismo. Esta función devuelve

como el fondo que corresponda al estado actual del clima con base en el código del mismo. Esta función devuelve una dupla, para que lo tengas en cuenta. Por otro lado, la función getFormattedDate recibe como parámetros una fecha y un formato específico, y devuelve un string que contiene dicha fecha formateada. Esta función será valiosa para desplegar las fechas o momentos esperados en los diferentes lugares de las vistas. Ten en cuenta que los formatos de fecha tienen una sintaxis específica. Por ejemplo, para formatear una fecha al siguiente formato: Martes, 3 de enero de 2017, 09:15:55 am, se requiere de la siguiente sintaxis: EEEE, d de MMMM de yyyy, hh:mm:ss a en donde EEEE es el formato de nombre de día completo, d es el número de día en formato de 1-2 dígitos, MMMM es el nombre completo de mes, yyyy es el número de año en formato 4 dígitos fijos, hh es el número de hora en lapsos de 12 horas y en formato de 2 dígitos fijos, mm es el número de minutos en formato de 2 dígitos fijos, ss es el número de segundos en formato de 2 dígitos fijos y a es el comodín para desplegar am o pm según el caso.

Ten en cuenta revisar detenidamente cada artefacto de software presente en el proyecto, intentando comprender qué hace o se supone debe hacer cada elemento y función. Debes prestar atención a los errores presentes para deducir qué elementos se esperan y cómo. Ten en cuenta los nombres de todos los elementos, y básate en ellos para determinar qué hacen o para qué se usan.

Ten en cuenta que requieres una API Key de OpenWeatherMap para poder hacer funcionar la aplicación. Sin esta API Key no podrás conectarte al servicio y no podrás obtener datos, por lo tanto la solución no podrá probarse y se tomará como reprobación inmediata. Para obtener tu propia API Key, deberás acceder a la ruta https://openweathermap.org/appid y seguir los pasos que allí se describen para registrarse, darse de alta en el sitio y obtener una API Key gratuita. Cuando todo esté listo, podrás acceder a tu cuenta y revisar la sección de API Keys para consultar las API Keys asignadas.



Ten en cuenta que puedes crear más de una pero al final todas suman y cuentan para el límite de consultas que poseen las cuentas gratuitas. Ten en cuenta que una API Key (incluyendo la primera de la cuenta) puede tomar un tiempo (aproximadamente 10 minutos o más) para ser activada y por lo tanto usable.

Descargar archivo base

Instrucciones para la entrega

Una vez finalices el proyecto, comprime la carpeta con los archivos y súbela al curso. Ten presentes los siguientes pasos para completar este último requerimiento, de la manera adecuada:

1. En una carpeta guarda tu proyecto y todos los recursos que consideres necesarios para su funcionamiento.
2. Utiliza la siguiente estructura para nombrar la carpeta Zip que entregarás pues nos permitirá identificarte: **NombreDelCurso_PrimerNombre_PrimerApellido.zip**.

3. Para subir el ZIP al curso, sigue los siguientes pasos:
    - Accede a la página **Evaluación Final**.
    - Haz clic en **Añadir envío** (hasta el final de la página).
    - Agrega una descripción en la sección Texto en línea y **Adjunta la CarpetaComprimida** en la sección Envíos de archivo.
    - Haz clic en **Guardar Cambios**.

Estatus de la entrega

Número de intento	Éste es el intento 1.					
Estatus de la entrega	Sin intento					
Estatus de calificación	No calificado					
Criterio para calificar	**Funcional**	Excelente "(1) - El proyecto puede abrirse y ejecutarse en cualquier simulador de iPhone desde XCode 8, cumpliendo con todos los lineamientos de ejecución requeridos para el proyecto. (2) - La aplicación no presenta controles adicionales ni carece de controles previamente definidos en el proyecto base. (3) - La aplicación consulta cualquier locación o región y presenta su	Muy Bien "(1) - El proyecto puede abrirse y ejecutarse en cualquier simulador de iPhone desde XCode 8, cumpliendo con todos los lineamientos de ejecución requeridos para el proyecto. (2) - La aplicación presenta controles adicionales y presenta todos los controles previamente definidos en el proyecto base. (3) - La aplicación consulta cualquier locación o región y	Satisfactorio "(1) - El proyecto puede abrirse y ejecutarse en cualquier simulador de iPhone desde XCode 8, cumpliendo con casi todos los lineamientos de ejecución requeridos para el proyecto. (2) - La aplicación presenta controles adicionales o deja de presentar controles previamente definidos en el proyecto base. (3) - La aplicación consulta cualquier locación o región y	Puede Mejorar "(1) - El proyecto puede abrirse y ejecutarse en cualquier simulador de iPhone desde XCode 8, cumpliendo algunos de los lineamientos de ejecución requeridos para el proyecto. (2) - La aplicación presenta controles adicionales o deja de presentar controles previamente definidos en el proyecto base. (3) - La aplicación consulta casi cualquier locación	Inadecuado "(1) - El proyecto no puede abrirse y ejecutarse en cualquier simulador de iPhone desde XCode 8, o cumpe muy pocos de los lineamientos de ejecución requeridos para el proyecto. (2) - La aplicación presenta controles adicionales o deja de presentar controles previamente definidos en el proyecto base. (3) - La aplicación no consulta la información del clima desde

	información del clima obtenida desde OpenWeatherMap conforme a la locación o región consultada. (4) - La aplicación no se bloquea ni se congela mientras hace las consultas o transporta la información y nunca se detiene ni se cierra abruptamente. La aplicación hace uso correcto del componente de indicador de actividad presente en el proyecto base." ***25 puntos***	presenta su información del clima obtenida desde OpenWeatherMap conforme a la locación o región consultada. (4) - La aplicación no se bloquea pero puede ocasionalmente congelarse mientras hace las consultas o transporta la información y nunca se detiene ni se cierra abruptamente. La aplicación hace uso correcto del componente de indicador de actividad presente en el proyecto base." ***20 puntos***	presenta su información del clima obtenida desde OpenWeatherMap conforme a la locación o región consultada. (4) - La aplicación ocasionalmente se bloquea o congela mientras hace las consultas o transporta la información pero no detiene su ejecución. La aplicación puede o no hacer uso correcto del componente de indicador de actividad presente en el proyecto base." ***15 puntos***	o región y presenta su información del clima obtenida desde OpenWeatherMap casi conforme a la locación o región consultada. (4) - La aplicación paulatinamente se bloquea o congela mientras hace las consultas o transporta la información e incluso detiene su ejecución. La aplicación puede o no hacer uso correcto del componente de indicador de actividad presente en el proyecto base." ***10 puntos***	OpenWeatherMap o la misma viene guardada dentro de la aplicación. (4) - La aplicación se bloquea o congela mientras hace las consultas o transporta la información o detiene su ejecución. La aplicación no usa el componente de indicador de actividad presente en el proyecto base." ***0 puntos***
Legibilidad y Calidad del código	Excelente "(1) - El proyecto presenta todos sus artefactos y elementos de software tanto física como conceptualmente diferenciados y delimitados conforme a su naturaleza, en uno de los cuatro grupos dispuestos en el proyecto base: Models, Views, Controllers, Utils. (3) - Cada clase, protocolo y estructura contiene la funcionalidad y contenido solicitados y no contienen funcionalidad ni contenido adicional. (4) - El código de todas los artefactos está correctamente indentado, con nomenclatura	Muy Bien "(1) - El proyecto presenta casi todos sus artefactos y elementos de software tanto física como conceptualmente diferenciados y delimitados conforme a su naturaleza, en uno de los cuatro grupos dispuestos en el proyecto base: Models, Views, Controllers, Utils. (3) - Cada clase, protocolo y estructura contiene la funcionalidad y contenido solicitados y pueden o no contener funcionalidad o contenido adicional. (4) - El código de casi todos los artefactos está correctamente	Satisfactorio "(1) - El proyecto presenta varios de sus artefactos y elementos de software tanto física como conceptualmente diferenciados y delimitados conforme a su naturaleza, en uno de los cuatro grupos dispuestos en el proyecto base: Models, Views, Controllers, Utils. (3) - Casi todas las clase, protocolos y estructuras contienen la funcionalidad y contenido solicitados y pueden o no contener funcionalidad o contenido adicional. (4) - El código de varios artefactos está correctamente	Puede Mejorar "(1) - El proyecto presenta pocos de sus artefactos y elementos de software tanto física como conceptualmente diferenciados y delimitados conforme a su naturaleza, en uno de los cuatro grupos dispuestos en el proyecto base: Models, Views, Controllers, Utils. (3) - Algunas clases, protocolos y estructuras contienen la funcionalidad y contenido solicitados y también contienen funcionalidad o contenido adicional. (4) - El código de varios artefactos presenta pocos errores de indentación, en	Inadecuado "(1) - El proyecto no presenta sus artefactos y elementos de software tanto física como conceptualmente diferenciados y delimitados conforme a su naturaleza, en uno de los cuatro grupos dispuestos en el proyecto base: Models, Views, Controllers, Utils. (3) - Muy pocas clases, protocolos y estructuras contienen la funcionalidad y contenido solicitados y también contienen funcionalidad o contenido adicional. (4) - El código de varios artefactos presenta varios errores de

	camel case y nombres en inglés y no contiene código sin usar. Los nombres asignados son descriptivos y coherentes a su propósito. " ***25 puntos***	indentado, con nomenclatura camel case y nombres en inglés y no contiene código sin usar. Los nombres asignados son descriptivos y coherentes a su propósito. " ***20 puntos***	indentado, con nomenclatura camel case y nombres en inglés y no contiene código sin usar. Los nombres en general son descriptivos y coherentes a su propósito. " ***15 puntos***	general con nomenclatura camel case y nombres en inglés y puede contener código sin usar. " ***10 puntos***	indentación, con poco uso de nomenclatura camel case y nombres en inglés y contiene código sin usar. " ***0 puntos***
Documentación	Excelente "(1) - Sigue todos los requerimientos y sugerencias dados para el proyecto. Demuestra un uso correcto y esperado de los diferentes patrones y orientaciones solicitados. (2) - Nombra de manera descriptiva y coherente todas las estructuras, protocolos, clases, métodos y propiedades, usando terminología en inglés y nomenclatura camel case. El proyecto presenta divisiones físicas de los elementos de Modelo, Vista, Controlador y Utilitarios. (3) - No presenta comentarios dentro del código, pues el mismo es claro, conciso y autodocumentado." ***25 puntos***	Muy Bien "(1) - Sigue casi todos los requerimientos y sugerencias dados para el proyecto. Demuestra en general un uso correcto y esperado de los diferentes patrones y orientaciones solicitados. (2) - Nombra de manera descriptiva y coherente todas las estructuras, protocolos, clases, métodos y propiedades, usando en general terminología en inglés y nomenclatura camel case. El proyecto presenta divisiones físicas de los elementos de Modelo, Vista, Controlador y Utilitarios. (3) - No presenta comentarios dentro del código, y en general es claro, conciso y autodocumentado." ***20 puntos***	Satisfactorio "(1) - Sigue varios de los requerimientos y sugerencias dados para el proyecto. Demuestra en general un uso correcto y esperado de los diferentes patrones y orientaciones solicitados. (2) - Nombra casi todos sus recursos de manera descriptiva y coherente, usando en general terminología en inglés y nomenclatura camel case. El proyecto puede o no presentar divisiones físicas de los elementos de Modelo, Vista, Controlador y Utilitarios. (3) - Presenta unos pocos comentarios dentro del código, y en general es claro, conciso y autodocumentado." ***15 puntos***	Puede Mejorar "(1) - Sigue varios de los requerimientos y sugerencias dados para el proyecto. Presenta algunas falencias en el uso correcto y esperado de los diferentes patrones y orientaciones solicitados. (2) - Nombra algunos de sus recursos de manera descriptiva y coherente, con poco uso de terminología en inglés y nomenclatura camel case. El proyecto puede o no presentar divisiones físicas de los elementos de Modelo, Vista, Controlador y Utilitarios. (3) - Presenta varios comentarios dentro del código, y en algunos casos el código no es claro ni autodocumentado." ***10 puntos***	Inadecuado "(1) - No sigue muchos de los requerimientos y sugerencias dados para el proyecto. Presenta muchas falencias en el uso y aplicación de los diferentes patrones y orientaciones solicitados. (2) - Nombra muy pocos recursos de manera descriptiva y coherente, con un uso casi nulo de terminología en inglés y nomenclatura camel case. El proyecto no presenta divisiones físicas de los elementos de Modelo, Vista, Controlador y Utilitarios. (3) - Presenta muchos comentarios dentro del código, y en en general el código no es claro ni autodocumentado." ***0 puntos***
Eficiencia de la solución planteada	Excelente "(1) - La aplicación se despliega perfectamente en todos los simuladores de iPhone conforme a todos los requerimientos. (2) - La aplicación	Muy Bien "(1) - La aplicación se despliega perfectamente en todos los simuladores de iPhone conforme a todos los requerimientos. (2) - La aplicación	Satisfactorio "(1) - La aplicación se despliega perfectamente en casi todos los simuladores de iPhone conforme a todos los requerimientos. (2) - La aplicación	Puede Mejorar "(1) - La aplicación se despliega en algunos de los simuladores de iPhone conforme a todos los requerimientos. (2) - La aplicación tarda en promedio	Inadecuado "(1) - La aplicación solo se despliega en un simulador de iPhone conforme a todos los requerimientos. (2) - La aplicación tarda en promedio más de cuatro

		tarda en promedio menos de dos minutos en compilar e instalar desde cero conforme a los requerimientos mínimos de hardware para XCode 8. Los despliegues en pantalla se hacen casi al instante que termina la consulta al servicio. (3) - Presenta un diseño limpio y las implementaciones de código son óptimas, sin redundancias y consulta una única vez por cada ciudad agregada. " ***25 puntos***	tarda en promedio dos minutos en compilar e instalar desde cero conforme a los requerimientos mínimos de hardware para XCode 8. Los despliegues en pantalla toman un poco más de lo que toma la consulta al servicio. (3) - Presenta un diseño en general limpio y las implementaciones de código son casi perfectas, sin redundancias y consulta una única vez por cada ciudad agregada. " ***20 puntos***	tarda en promedio menos de tres minutos en compilar e instalar desde cero conforme a los requerimientos mínimos de hardware para XCode 8. La consulta o los despliegues en pantalla toman un poco más de lo que toma la consulta al servicio. (3) - Presenta un diseño en general limpio y las implementaciones de código son en genral muy buenas, sin redundancias y consulta una única vez por cada ciudad agregada. " ***15 puntos***	cuatro minutos en compilar e instalar desde cero conforme a los requerimientos mínimos de hardware para XCode 8. La consulta o los despliegues en pantalla toman al rededor de un minuto luego de la terminación de la consulta. (3) - Presenta un diseño algo desordenado pero las implementaciones de código son en genral muy buenas, con algunas redundancias y consulta una única vez por cada ciudad agregada. " ***10 puntos***	minutos en compilar e instalar desde cero conforme a los requerimientos mínimos de hardware para XCode 8. Los despliegues en pantalla toman más de un minuto luego de la terminación de la consulta. (3) - Presenta un diseño en general desordenado y las implementaciones de código son en genral muy buenas, con varias redundancias. (4) Consulta varias veces la misma ciudad luego de ser agregada. " ***0 puntos***

Última modificación	-

Añadir envío

Hacer cambios a su envío