|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Sam Ruby | **Editorial** | Pragmatic Bookshelf |
| **Título** | Agile Web Development with Rails 5 | **Ciudad, país** | Estados Unidos, Raleigh, North Carolina |
| **Año** | 2016 |  | |
| **Tema: Patrón de Diseño MVC**  **p.42**  “The model is responsible for maintaining the state of the application. Sometimes this state is transient, lasting for just a couple of interactions with the user. Sometimes the state is permanent and is stored outside the application, often in a database. A model is more than data; it enforces all the business rules that apply to that data. For example, if a discount shouldn’t be applied to orders of less than $20, the model enforces the constraint. This makes sense; by putting the implementation of these business rules in the model, we make sure that nothing else in the application can make our data invalid. The model acts as both a gatekeeper and a data store. The view is responsible for generating a user interface, normally based on data in the model. For example, an online store has a list of products to be displayed on a catalog screen. This list is accessible via the model, but it’s a view that formats the list for the end user. Although the view might present the user with various ways of inputting data, the view itself never handles incoming data. The view’s work is done once the data is displayed. There may well be many views that access the same model data, often for different purposes. The online report erratum • discuss store has a view that displays product information on a catalog page, and another set of views used by administrators to add and edit products. Controllers orchestrate the application. Controllers receive events from the outside world (normally, user input), interact with the model, and display an appropriate view to the user.” | | | |
| **Edición** | Primera edición | **Numero de ficha bibliográfica** | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Monte Galiano, J | **Editorial** | Ednmnal UOC (Oberta UOC Pubhshmg, SL) |
| **Título** | Implantar scrum con éxito | **Ciudad, país** | Barcelona, España |
| **Año** | 2016 |  | |
| **Tema: Scrum**  **p.21**  Scrum esta basado, por un lado, en la teoría del control empírico de procesos para la gestión de sistemas adaptativos complejos Los tres pilares de este proceso son los siguientes: ' Transparencia: los aspectos signiﬁcativos del proceso tienen que ser conocidos por todo aquel que participa, lo cual conlleva que estos aspectos estén deﬁnidos mediante un estándar común, de forma que todo el mundo tenga la misma percepción de las características de cada aspecto (por ejemplo, la deﬁnición de grabado). ' Inspección: todo proceso persigue un objetivo y, para llei gar a ese objetivo, hace falta que los parúcipantes en el proceso evalúen de manera continua sus resultados, y el proceso mismo, para detectar posibles desviaciones tan pronto como sea posible. ' Adaptación: cuando se detecta una desviación, la respuesi ta debe ser la adaptación; es decir, la adopción de acciones o planes que, o bien ayuden a corregir la desviación, o bien reconﬁguren el objetivo | | | |
| **Edición** | Primera edición | **Numero de ficha bibliográfica** | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Ian Sommerville | **Editorial** | Pearson educacion Mexico |
| **Título** | Ingenieria de Software | **Ciudad, país** | Juarez,Mexico |
| **Año** | 2011 |  | |
| **Tema: Ingenieria de Software**  **p.35**  La ingeniería de software (el término es discutido por cuanto el desarrollo de software no es en muchas ocasiones considerado como una ingeniería) es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software,​ y el estudio de estos enfoques, es decir, el estudio de las aplicaciones de la ingeniería al software. Integra matemáticas, ciencias de la computación y prácticas cuyos orígenes se encuentran en la ingeniería.  La creación del software es un proceso intrínsecamente creativo y la ingeniería del software trata de sistematizar este proceso con el fin de acotar el riesgo de fracaso en la consecución del objetivo, por medio de diversas técnicas que se han demostrado adecuadas sobre la base de la experiencia previa. | | | |
| **Edición** | 9na edición | **Numero de ficha bibliográfica** | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL : <https://www.atlassian.com/git/tutorials/why-git> | |
| **Tema: GIT**  **p.35**  Git is the most commonly used version control system today and is quickly becoming the standard for version control. Git is a distributed version control system, meaning your local copy of code is a complete version control repository. These fully-functional local repositories make it is easy to work offline or remotely. You commit your work locally, and then sync your copy of the repository with the copy on the server. This paradigm differs from centralized version control where clients must synchronize code with a server before creating new versions of code.  Git’s flexibility and popularity make it a great choice for any team. Many developers and college graduates already know how to use Git. Git’s user community has created many resources to train developers and Git’s popularity make it easy to get help when you need it. Nearly every development environment has Git support and Git command line tools run on every major operating system.  Unlike some version control software, Git is not fooled by the names of the files when determining what the storage and version history of the file tree should be, instead, Git focuses on the file content itself. After all, source code files are frequently renamed, split, and rearranged. The object format of Git's repository files uses a combination of delta encoding (storing content differences), compression and explicitly stores directory contents and version metadata objects.  Security  Git has been designed with the integrity of managed source code as a top priority. The content of the files as well as the true relationships between files and directories, versions, tags and commits, all of these objects in the Git repository are secured with a cryptographically secure hashing algorithm called SHA1. This protects the code and the change history against both accidental and malicious change and ensures that the history is fully traceable.  With Git, you can be sure you have an authentic content history of your source code.  Some other version control systems have no protections against secret alteration at a later date. This can be a serious information security vulnerability for any organization that relies on software development.  Flexibility  One of Git's key design objectives is flexibility. Git is flexible in several respects: in support for various kinds of nonlinear development workflows, in its efficiency in both small and large projects and in its compatibility with many existing systems and protocols.  Git has been designed to support branching and tagging as first-class citizens (unlike SVN) and operations that affect branches and tags (such as merging or reverting) are also stored as part of the change history. Not all version control systems feature this level of tracking. | | | |
|  | | **Numero de ficha bibliográfica** | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL : <https://es.reactjs.org/> | |
| **Tema: React**  React is a popular library used to create user interfaces. It was built at Facebook to address some of the challenges associated with large-scale, data-driven websites. When React was released in 2013, the project was initially viewed with some skepti‐ cism because the conventions of React are quite unique. In an attempt to not intimidate new users, the core React team wrote an article called “Why React?” that recommended that you “Give It [React] Five Minutes.” They wanted to encourage people to work with React first before thinking that their approach was too crazy. Yes, React is a small library that doesn’t come with everything you might need out of the box to build your application. Give it five minutes. Yes, in React, you write code that looks like HTML right in your JavaScript. And yes, those tags require preprocessing to run in a browser. And you’ll probably need a build tool like webpack for that. Give it five minutes. If you read that article—as we did—you may have been dazzled by the promise of a new JavaScript library—a library that would solve all of our problems with the DOM; a library that would always be easy to work with and would never hurt us. Then the questions start to arise: how do I convert this JSX? How do I load data? Where does the CSS go? What is declarative programming? Every path leads to more questions about how to incorporate this library in your actual day to day work. Every conversation introduces new terminology, new techniques, and more questions. | | | |
|  | | **Numero de ficha bibliográfica** | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL :<https://es.redux.js.org/> | |
| **Tema: Redux**  Redux es un contenedor predecible del estado de aplicaciones JavaScript.  Te ayuda a escribir aplicaciones que se comportan de manera consistente, corren en distintos ambientes (cliente, servidor y nativo), y son fáciles de probar. Además de eso, provee una gran experiencia de desarrollo, gracias a edición en vivo combinado con un depurador sobre una línea de tiempo.  Puedes usar Redux combinado con React, o cual cualquier otra librería de vistas. Es muy pequeño (2kB) y no tiene dependencias.  Redux has emerged as one of the clear winners in the field of Flux or Flux-like libra‐ ries. Redux is based on Flux, and it was designed to tackle the challenge of under‐ standing how data changes flow through your application. Redux was developed by Dan Abramov and Andrew Clark. Since creating Redux, both have been hired by Facebook to work on the React team. Andrew Clark was working on version 4 of Flummox, another Flux-based library, when he started assisting Dan with the task of completing Redux. The message on the npm page for Flummox reads: Eventually 4.x should be the last major release but it never happened. If you want the latest features, then use Redux instead. It’s really great.1 Redux is surprisingly small, only 99 lines of code. We have mentioned that Redux is Flux-like, but it is not exactly Flux. It has actions, action creators, a store, and action objects that are used to change state. Redux sim‐ plifies the concepts of Flux a bit by removing the dispatcher, and representing appli‐ cation state with a single immutable object. Redux also introduces reducers, which are not a part of the Flux pattern. Reducers are pure functions that return a new state based on the current state and an action: (state, action) => newState. | | | |
|  | | **Numero de ficha bibliográfica** | 6 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL : <https://facebook.github.io/react-native/docs/tutorial> | |
| **Tema: React native**  React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. It’s based on React, Facebook’s JavaScript library for building user interfaces, but instead of targeting the browser, it targets mobile platforms. In other words: web developers can now write mobile applications that look and feel truly “native,” all from the comfort of a JavaScript library that we already know and love. Plus, because most of the code you write can be shared between platforms, React Native makes it easy to simultaneously develop for both Android and iOS.  Similar to React for the Web, React Native applications are written using a mixture of JavaScript and XML-esque markup, known as JSX. Then, under the hood, the React Native “bridge” invokes the native rendering APIs in Objective-C (for iOS) or Java (for Android). Thus, your application will render using real mobile UI components, not webviews, and will look and feel like any other mobile application. React Native also exposes JavaScript interfaces for platform APIs, so your React Native apps can access platform features like the phone camera, or the user’s location.  React Native currently supports both iOS and Android, and has the potential to expand to future platforms as well. In this book, we’ll cover both iOS and Android. The vast majority of the code we write will be cross-platform. And yes: you can really use React Native to build production-ready mobile applications! Some anecdota: Facebook, Palantir, and TaskRabbit are already using it in production for user-facing applications.  Advantages of React Native  The fact that React Native actually renders using its host platform’s standard rendering APIs enables it to stand out from most existing methods of cross-platform application development, like Cordova or Ionic. Existing methods of writing mobile applications using combinations of JavaScript, HTML, and CSS typically render using webviews. While this approach can work, it also comes with drawbacks, especially around performance. Additionally, they do not usually have access to the host platform’s set of native UI elements. When these frameworks do try to mimic native UI elements, the results usually “feel” just a little off; reverse-engineering all the fine details of things like animations takes an enormous amount of effort, and they can quickly become out of date.  In contrast, React Native actually translates your markup to real, native UI elements, leveraging existing means of rendering views on whatever platform you are working with. Additionally, React works separately from the main UI thread, so your application can maintain high performance without sacrificing capability. The update cycle in React Native is the same as in React: when props or state change, React Native re-renders the views. The major difference between React Native and React in the browser is that React Native does this by leveraging the UI libraries of its host platform, rather than using HTML and CSS markup.  For developers accustomed to working on the Web with React, this means you can write mobile apps with the performance and look and feel of a native application, | | | |
|  | | **Numero de ficha bibliográfica** | 7 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL :<https://flutter-es.io/> | |
| **Tema: Flutter**  Flutter es el kit de herramientas de UI de Google para realizar hermosas aplicaciones, compiladas nativamente, para móvil, web y escritorio desde una única base de código.  Develop for iOS and Android from a single codebase  Do more with less code, even on a single OS, with a modern, expressive language and a declarative approach  Prototype and iterate easily  Experiment by changing code and reloading as your app runs (with hot reload)  Fix crashes and continue debugging from where the app left off  Create beautiful, highly-customized user experiences  Benefit from a rich set of Material Design and Cupertino (iOS-flavor) widgets built using Flutter’s own framework  Realize custom, beautiful, brand-driven designs, without the limitations of OEM widget sets  Core principles  Flutter includes a modern react-style framework, a 2D rendering engine, ready-made widgets, and development tools. These components work together to help you design, build, test, and debug apps. Everything is organized around a few core principles.  Everything’s a widget  Widgets are the basic building blocks of a Flutter app’s user interface. Each widget is an immutable declaration of part of the user interface. Unlike other frameworks that separate views, view controllers, layouts, and other properties, Flutter has a consistent, unified object model: the widget.  A widget can define:  a structural element (like a button or menu)  a stylistic element (like a font or color scheme)  an aspect of layout (like padding)  and so on…  Widgets form a hierarchy based on composition. Each widget nests inside, and inherits properties from, its parent. There is no separate “application” object. Instead, the root widget serves this role.  You can respond to events, like user interaction, by telling the framework to replace a widget in the hierarchy with another widget. The framework then compares the new and old widgets and efficiently updates the user interface.  Composition > inheritance  Widgets are themselves often composed of many small, single-purpose widgets that combine to produce powerful effects. For example, Container, a commonly-used widget, is made up of several widgets responsible for layout, painting, positioning, and sizing. Specifically, Container is made up of LimitedBox, ConstrainedBox, Align, Padding, DecoratedBox, and Transform widgets. Rather than subclassing Container to produce a customized effect, you can compose these, and other, simple widgets in novel ways.  The class hierarchy is shallow and broad to maximize the possible number of combinations. | | | |
|  | | **Numero de ficha bibliográfica** | 8 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Pryyesh patel |  | |
| **Título** | Hybrid vs native |
| **Fecha** | 04/12/2018 | URL :<https://blog.techmagic.co/native-vs-hybrid-apps/> | |
| **Tema: Hybrid vs native**  Native apps  Native apps are written in a programming language specific to the platform they’re being developed for. This would typically be Objective-C or Swift for iOS and Java for Android. Native apps typically have better performance with rendering and animations than hybrid apps.  Hybrid apps  A hybrid app is a mobile app that contains a web view (essentially an isolated browser instance) to run a web application inside of a native app, using a native app wrapper that can communicate with the native device platform and the web view. This means web applications can run on a mobile device and have access to the device, such as the camera or GPS features.  Hybrid apps are possible because of tools that have been created that facilitate the communication between the web view and the native platform. These tools are not part of the official iOS or Android platforms, but are third party tools such as Apache Cordova, which is used in this book. When a hybrid app is built, it will be compiled, transforming your web application into a native app.  There are so many hybrid mobile frameworks such as Ionic, NativeScript, React Native, Xamarin, PhoneGap etc. We decided to compare native apps with top 2 hybrid frameworks — Ionic and React Native.  Ionic  Ionic is a framework that essentially allows developers to create hybrid mobile apps using web technologies like HTML, CSS and JavaScript.  A hybrid mobile app is built using technologies typically used for the web. Hybrid apps are hosted inside native applications that allow them to access the device’s camera, pedometer and other functionalities,removing the need to develop for any specific device or operating system.  This basically means that you are creating a website wrapped up inside an app.  React Native  React Native is a framework developed by Facebook for creating native-style apps for iOS & Android under one common language, JavaScript.  Unlike hybrid apps, native apps are built especially for the platform they’re to be used on (iOS,Android etc). React Native allows a proportion of the code to be shared between platforms and empowers developers to create apps which feel less clunky and perform better than hybrid apps.  With both React Native and Native apps, gestures like pinching or double tapping work in the way you’d expect them to work on your operating system.  We made table with comparing each technology and | | | |
|  | | **Numero de ficha bibliográfica** | 9 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL :<https://www.xplenty.com/blog/the-sql-vs-nosql-difference/> | |
| **Tema: SQL vs No SQL**  The Big Picture Differences Between SQL and NoSQL  The Language  Think of a town - we’ll call it Town A - where everyone speaks the same language. All of the businesses are built around it, every form of communication uses it. In short, it’s the only way that the residents understand and interact with the world around them. Changing that language in one place would be confusing and disruptive for everyone.  Now, think of another town, Town B, where every home can speak a different language. Everyone interacts with the world differently, and there’s no “universal” understanding or set organization. If one home is different, it doesn’t affect anyone else at all.  This helps illustrate one of the fundamental differences between SQL (relational) and NoSQL (non-relational) databases, and this distinction has big implications. Let’s explain:  SQL databases: SQL databases use structured query language (SQL) for defining and manipulating data. On one hand, this is extremely powerful: SQL is one of the most versatile and widely-used options available, making it a safe choice and especially great for complex queries. On the other hand, it can be restrictive. SQL requires that you use predefined schemas to determine the structure of your data before you work with it. In addition, all of your data must follow the same structure. This can require significant up-front preparation, and, as with Town A, it can mean that a change in the structure would be both difficult and disruptive to your whole system.  NoSQL databases: NoSQL databases, on the other hand, have dynamic schemas for unstructured data, and data is stored in many ways: They can be column-oriented, document-oriented, graph-based or organized as a KeyValue store. This flexibility means that:  The Scalability  In most situations, SQL databases are vertically scalable, which means that you can increase the load on a single server by increasing things like CPU, RAM or SSD. NoSQL databases, on the other hand, are horizontally scalable. This means that you handle more traffic by sharding, or adding more servers in your NoSQL database. It’s like adding more floors to the same building versus adding more buildings to the neighborhood. The latter can ultimately become larger and more powerful, making NoSQL databases the preferred choice for large or ever-changing data sets.  The Structure  SQL databases are table-based, while NoSQL databases are either document-based, key-value pairs, graph databases or wide-column stores. This makes relational SQL databases a better option for applications that require multi-row transactions - such as an accounting system - or for legacy systems that were built for a relational structure.  Some examples of SQL databases include MySQL, Oracle, PostgreSQL, and Microsoft SQL Server. NoSQL database examples include MongoDB, BigTable, Redis, RavenDB Cassandra, HBase, Neo4j and CouchDB. | | | |
|  | | **Numero de ficha bibliográfica** | 10 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL :<https://www.postgresql.org/about/> | |
| **Tema: Postgres**  PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the POSTGRES project at the University of California at Berkeley and has more than 30 years of active development on the core platform.  PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on all major operating systems, has been ACID-compliant since 2001, and has powerful add-ons such as the popular PostGIS geospatial database extender. It is no surprise that PostgreSQL has become the open source relational database of choice for many people and organisations.  Getting started with using PostgreSQL has never been easier - pick a project you want to build, and let PostgreSQL safely and robustly store your data.  PostgreSQL comes with many features aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help you manage your data no matter how big or small the dataset. In addition to being free and open source, PostgreSQL is highly extensible. For example, you can define your own data types, build out custom functions, even write code from different programming languages without recompiling your database!  PostgreSQL tries to conform with the SQL standard where such conformance does not contradict traditional features or could lead to poor architectural decisions. Many of the features required by the SQL standard are supported, though sometimes with slightly differing syntax or function. Further moves towards conformance can be expected over time. As of the version 12 release in October 2019, PostgreSQL conforms to at least 160 of the 179 mandatory features for SQL:2016 Core conformance. As of this writing, no relational database meets full conformance with this standard. | | | |
|  | | **Numero de ficha bibliográfica** | 11 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Monte Galiano, J | **Editorial** | Ednmnal UOC (Oberta UOC Pubhshmg, SL) |
| **Título** | El Proceso  Unificado de  Desarrollo de  Software | **Ciudad, país** | Barcelona, España |
| **Año** | 2016 |  | |
| **Tema: Proceso unificado**  **p.21**  El Proceso Unificado es un proceso de desarrollo de software: “conjunto de  actividades necesarias para transformar los requisitos del usuario en un  sistema software”.  · RUP es un marco genérico que puede especializarse para una variedad de  tipos de sistemas, diferentes áreas de aplicación, tipos de organizaciones,  niveles de aptitud y diferentes tamaños de proyectos.  · RUP está basado en componentes. El sw esta formado por componentes  software interconectados a través de interfaces.  · RUP está dirigido por casos de uso, centrado en la arquitectura, y es  iterativo e incremental.  Dirigido por Casos de Uso  · Un caso de uso es un fragmento de funcionalidad del sistema que proporciona  un resultado de valor a un usuario. Los casos de uso modelan los  requerimientos funcionales del sistema.  · Todos los casos de uso juntos constituyen el modelo de casos de uso.  · Los casos de uso también guían el proceso de desarrollo (diseño,  implementación, y prueba). Basándose en los casos de uso los desarrolladores  crean una serie de modelos de diseño e implementación que llevan a cabo los  casos de uso. De este modo los casos de uso no solo inician el proceso de  desarrollo sino que le proporcionan un hilo conductor, avanza a través de una  serie de flujos de trabajo que parten de los casos de uso.  Centrado en la Arquitectura  La arquitectura de un sistema software se describe mediante diferentes vistas del  sistema en construcción.  El concepto de arquitectura software incluye los aspectos estáticos y dinámicos más  significativos del sistema.  La arquitectura es una vista del diseño completo con las características más  importantes resaltadas, dejando los detalles de lado. | | | |
| **Edición** | Primera edición | **Numero de ficha bibliográfica** | 12 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Juan Sebastián Villanueva |  | |
| **Título** | SCRUM Y RUP: COMPARATIVA Y PROPUESTA METODOLÓGICA |
| **Fecha** | 05/03/2018 | URL :<https://flutter-es.io/> | |
| **Tema: Scrum vs RUP**  Ambas metodologías tienen sus limitaciones y debilidades, así como las metodologías ágiles son las más adecuadas para proyectos pequeños y medianos, no son las más adecuadas para sistemas de gran escala que requieran de interacciones complejas con otros sistemas, debido a que estos sistemas requieren de un nivel de precisión bastante alto y tienen un gran riesgo de construcción. No sería conveniente implementar una metodología ágil para el desarrollo de un sistema crítico en el cual es necesario el análisis detallado de todos los requerimientos para comprender su complejidad e implicaciones, debido a la complejidad y la extrema precisión quepueda tener la captura de requerimientos, en los cuáles las metodologías agiles como SCRUM ofrecen demasiada flexibilidad.Karlstrm y Runeson [9] encontraron que los métodos agiles proveen herramientas ponderosas para la planeación a pequeña escala, control del trabajo diario, reporte de progreso y la mejora en los canales de comunicación del equipo.Los procesos de desarrollo agiles son una gran opción cuando el objetivo es el incremento de la productividad [10], [11] ya que se enfoca en la importancia del manejo del equipo y de personas, o el mejoramiento de la capacidad de respuesta a peticiones de cambio hechas a través del ciclo de desarrollo de software [12]. | | | |
|  | | **Numero de ficha bibliográfica** | 13 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL https://www.redhat.com/es/topics/containers/what-is-docker | |
| **Tema: Docker**  La palabra "DOCKER" se refiere a varias cosas. Esto incluye un proyecto de la comunidad open source; las herramientas del proyecto open source; Docker Inc., la empresa que es la principal promotora de ese proyecto; y las herramientas que la empresa admite formalmente. El hecho de que las tecnologías y la empresa compartan el mismo nombre puede ser confuso.  A continuación, le presentamos una breve explicación:  "Docker", el software de TI, es una tecnología de creación de contenedores que permite la creación y el uso de contenedores de Linux®.  La comunidad open source Docker trabaja para mejorar estas tecnologías a fin de beneficiar a todos los usuarios de forma gratuita.  La empresa, Docker Inc., desarrolla el trabajo de la comunidad Docker, lo hace más seguro y comparte estos avances con el resto de la comunidad. También respalda las tecnologías mejoradas y reforzadas para los clientes empresariales.  La tecnología Docker usa el kernel de Linux y las funciones de este, como Cgroups y namespaces, para segregar los procesos, de modo que puedan ejecutarse de manera independiente. El propósito de los contenedores es esta independencia: la capacidad de ejecutar varios procesos y aplicaciones por separado para hacer un mejor uso de su infraestructura y, al mismo tiempo, conservar la seguridad que tendría con sistemas separados.  Las herramientas del contenedor, como Docker, ofrecen un modelo de implementación basado en imágenes. Esto permite compartir una aplicación, o un conjunto de servicios, con todas sus dependencias en varios entornos. Docker también automatiza la implementación de la aplicación (o conjuntos combinados de procesos que constituyen una aplicación) en este entorno de contenedores.  Estas herramientas desarrolladas a partir de los contenedores de Linux, lo que hace a Docker fácil de usar y único, otorgan a los usuarios un acceso sin precedentes a las aplicaciones, la capacidad de implementar rápidamente y control sobre las versiones y su distribución. | | | |
|  | | **Numero de ficha bibliográfica** | 14 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL <https://www.freecodecamp.org/news/an-introduction-to-the-flux-architectural-pattern-674ea74775c9/> | |
| **Tema: Flux**  Flux is an architectural pattern proposed by Facebook for building SPAs. It suggests to split the application into the following parts:  Stores  Dispatcher  Views  Action / Action Creators  Store  Store manages the state. It can store both domain state and user interface state.  Store and state are different concepts. State is the data value. Store is a behavior object that manages state through methods. In the case of managing books: the book list is the state and BookStore manages that list.  A store manages multiple objects. It is the single source of truth in regards to those specific objects. In an application there can be many stores. For example: BookStore, AuthorStore, UserStore.  There are no setter methods on the store. You can only request state change by passing an action to the dispatcher.  A store listens for all actions and decides on which of them to act. This usually means a switch statement. Once the store has made the state changes, it will emit a change event. The store is an event emitter.  Stores don’t take other stores as dependencies.  Dispatcher  Dispatcher is a single object that broadcasts actions/events to all registered stores. Stores need to register for events when the application starts.  When an action comes in, it will pass that action to all registered stores.  View  View is the user interface component. It is responsible for rendering the user interface and for handling the user interaction. Views are in a tree structure.  Views listen for store changes and re-render.  Views can be further split in Presentation and Container Views. | | | |
|  | | **Numero de ficha bibliográfica** | 15 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL :<https://medium.com/javascript-scene/master-the-javascript-interview-what-is-functional-programming-7f218c68b3a0> | |
| **Tema: Functional Programmming**  Functional programming is a programming paradigm, meaning that it is a way of thinking about software construction based on some fundamental, defining principles (listed above). Other examples of programming paradigms include object oriented programming and procedural programming.  Functional code tends to be more concise, more predictable, and easier to test than imperative or object oriented code — but if you’re unfamiliar with it and the common patterns associated with it, functional code can also seem a lot more dense, and the related literature can be impenetrable to newcomers.  If you start googling functional programming terms, you’re going to quickly hit a brick wall of academic lingo that can be very intimidating for beginners. To say it has a learning curve is a serious understatement. But if you’ve been programming in JavaScript for a while, chances are good that you’ve used a lot of functional programming concepts & utilities in your real software.  Pure functions have lots of properties that are important in functional programming, including referential transparency (you can replace a function call with its resulting value without changing the meaning of the program). Read “What is a Pure Function?” for more details.  Function composition is the process of combining two or more functions in order to produce a new function or perform some computation. For example, the composition f . g (the dot means “composed with”) is equivalent to f(g(x)) in JavaScript. Understanding function composition is an important step towards understanding how software is constructed using the functional programming. Read “What is Function Composition?” for more.  Shared State  Shared state is any variable, object, or memory space that exists in a shared scope, or as the property of an object being passed between scopes. A shared scope can include global scope or closure scopes. Often, in object oriented programming, objects are shared between scopes by adding properties to other objects. | | | |
|  | | **Numero de ficha bibliográfica** | 16 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Sam Ruby | **Editorial** | Pragmatic Bookshelf |
| **Título** | Agile Web Development with Rails 5 | **Ciudad, país** | Estados Unidos, Raleigh, North Carolina |
| **Año** | 2016 |  | |
| **Tema: Rails**  **p.42**  Ruby on Rails is a framework that makes it easier to develop, deploy, and maintain web applications. During the ten years since its initial release, Rails went from being an unknown toy to being a worldwide phenomenon; more important, it has become the framework of choice for the implementation of a wide range of applications. Why is that? Rails Simply Feels Right A large number of developers were frustrated with the technologies they were using to create web applications. It didn’t seem to matter whether they used Java, PHP, or .NET—there was a growing sense that their jobs were just too damn hard. And then, suddenly, along came Rails, and Rails was easier. But easy on its own doesn’t cut it. We’re talking about professional developers writing real-world websites. They wanted to feel that the applications they were developing would stand the test of time—that they were designed and implemented using modern, professional techniques. So, these developers dug into Rails and discovered it wasn’t just a tool for hacking out sites. For example, all Rails applications are implemented using the Model-ViewController (MVC) architecture. Java developers are used to frameworks such as Tapestry and Struts, which are based on MVC. But Rails takes MVC further: when you develop in Rails, you start with a working application, there’s a place for each piece of code, and all the pieces of your application interact in a standard way. Professional programmers write tests. And again, Rails delivers. All Rails applications have testing support baked right in. As you add functionality to the code, Rails automatically creates test stubs for that functionality. The framework makes it easy to test applications, and as a result, Rails applications tend to get tested. | | | |
| **Edición** | Primera edición | **Numero de ficha bibliográfica** | 17 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL :<https://flutter-es.io/> | |
| **Tema: Firebase**  Firebase is a Backend-as-a-Service — BaaS — that started as a YC11 startup and grew up into a next-generation app-development platform on Google Cloud Platform.  I’ve been using Firebase since 2013. I was late to the game :)  I use Firebase to power my family’s business, Calligraphy.org. I also use it for everything I’ve developed in the last three years.  Firebase frees developers to focus crafting fantastic user experiences. You don’t need to manage servers. You don’t need to write APIs. Firebase is your server, your API and your datastore, all written so generically that you can modify it to suit most needs. Yeah, you’ll occasionally need to use other bits of the Google Cloud for your advanced applications. Firebase can’t be everything to everybody. But it gets pretty close.  It’s a Realtime Database  Real-time data is the way of the future. Nothing compares to it.  Most databases require you to make HTTP calls to get and sync your data. Most databases give you data only when you ask for it.  When you connect your app to Firebase, you’re not connecting through normal HTTP. You’re connecting through a WebSocket. WebSockets are much, much faster than HTTP. You don’t have to make individual WebSocket calls, because one socket connection is plenty. All of your data syncs automagically through that single WebSocket as fast as your client’s network can carry it.  Firebase sends you new data as soon as it’s updated. When your client saves a change to the data, all connected clients receive the updated data almost instantly.  It’s File Storage  Firebase Storage provides a simple way to save binary files — most often images, but it could be anything — to Google Cloud Storage directly from the client!!!  Firebase Storage has it’s own system of security rules to protect your GCloud bucket from the masses, while granting detailed write privileges to your authenticated clients.  It’s Authentication  Firebase auth has a built in email/password authentication system. It also supports OAuth2 for Google, Facebook, Twitter and GitHub. We’ll focus on email/password authentication for the most part. Firebase’s OAuth2 system is well-documented and mostly copy/paste.  If you’ve ever written an authentication system, let’s commiserate for a moment. Custom authentication is terrible. I will never write an auth system again for as long as I live. I fell in love with Firebase Auth at first sight, and the flame has never wavered. Sometimes I get frustrated. Sometimes we fight. But I never forget the cold, dark abyss of a custom auth system. I count my blessings.  It’s Hosting  And to make your development particularly painless, Firebase hosting utilizes Superstatic, which you can run locally for all of your testing. I run Superstatic as BrowserSync middleware. The following implementation uses Gulp, but Gulp is purely optional. | | | |
|  | | **Numero de ficha bibliográfica** | 18 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL :<https://es.redux.js.org/> | |
| **Tema: Google Maps**  **Uno más de los productos llevados al mercado por el gigante Google Inc., mismo que**  **anunciaba en Google Blog el 8 de febrero del 2005 el inicio y puesta en marcha de sus**  **mapas digitales, soportado en una primera fase solo por los usuarios de Internet Explorer**  **y Mozilla Firefox, agregándose el soporte para Opera y Safari el 25 de febrero del mismo**  **año.**  **Se trata de un servidor de aplicaciones de mapas en la Web (Internet), con la capacidad de**  **hacer acercamientos o alejamientos (Zoom) al mapa, controlando con el mouse o las**  **teclas de dirección los movimientos para encontrar la ubicación que se desee; además los**  **usuarios pueden ingresar una dirección, una intersección o un área en general para buscar**  **en el mapa y encontrar los resultados.**  **Google Maps, permite la creación de pasos para llegar a alguna dirección creando una**  **lista paso a paso para saber el cómo llegar a su destino, calculando el tiempo necesario y**  **la distancia recorrida entre las ubicaciones.**  **Google Maps puso a disposición de los desarrolladores sus códigos fuentes llamados**  **APIS, los mismos que permiten introducir los mapas de Google Maps en cualquier**  **aplicación con el uso de su codificación y con ello se pueden aplicar nuevas formas de ver**  **el mundo.**  **Con la innovación de las herramientas de búsqueda y el movimiento en el mapa de**  **Google Maps, se ha incrementado el interés en el uso de las imágenes satelitales, se han**  **creado sitios que buscan crear la base de datos de lugares conocidos y vistos desde el**  **29**  **espacio, como por ejemplo: estadios, construcciones antiguas, carreteras, edificios, ciclo**  **vías, etc.** | | | |
|  | | **Numero de ficha bibliográfica** | 19 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Sam Ruby | **Editorial** | Pragmatic Bookshelf |
| **Título** | Agile Web Development with Rails 5 | **Ciudad, país** | Estados Unidos, Raleigh, North Carolina |
| **Año** | 2016 |  | |
| **Tema: Ruby**  **p.42**  Ruby Is an Object-Oriented Language Everything you manipulate in Ruby is an object, and the results of those manipulations are themselves objects. When you write object-oriented code, you’re normally looking to model concepts from the real world. Typically, during this modeling process you discover categories of things that need to be represented. In an online store, the concept of a line item could be such a category. In Ruby, you’d define a class to represent each of these categories. You then use this class as a kind of factory that generates objects—instances of that class. An object is a combination of state (for example, the quantity and the product ID) and methods that use that state (perhaps a method to calculate the line item’s total cost). We’ll show how to create classes in Classes, on page 56. report erratum • discuss You create objects by calling a constructor, a special method associated with a class. The standard constructor is called new(). Given a class called LineItem, you could create line item objects as follows: line\_item\_one = LineItem.new line\_item\_one.quantity = 1 line\_item\_one.sku = "AUTO\_B\_00" You invoke methods by sending a message to an object. The message contains the method’s name, along with any parameters the method may need. When an object receives a message, it looks into its own class for a corresponding method. Let’s look at some method calls: "dave".length line\_item\_one.quantity() cart.add\_line\_item(next\_purchase) submit\_tag "Add to Cart" Parentheses are generally optional in method calls. In Rails applications, you’ll find that most method calls involved in larger expressions have parentheses, while those that look more like commands or declarations tend not to have them. Methods have names, as do many other constructs in Ruby. Names in Ruby have special rules—rules that you may not have seen if you come to Ruby from another language. | | | |
| **Edición** | Primera edición | **Numero de ficha bibliográfica** | 20 |