

Unidad 4 - Introducción al panel de AzureDevops

NeonCode, la empresa al cargo del mantenimiento y desarrollo de TheGoodHome, el portal web de inmobiliaria y alquiler de viviendas, acaba de aterrizar en Azure DevOps.

En una reunión únicamente dedicada para ello, muestran por pantalla el portal web para que todo el equipo pueda descubrir todo lo que permite este servicio.

Creación de una organización

El primer paso para comenzar a operar en Azure DevOps es la creación de una organización. Una organización puede ser entendida como un proyecto a entender, un grupo de personas dedicadas con un mismo objetivo.

Puedes tener tantas organizaciones como necesites, pero con cuidado no dividir tanto como para que sea incómodo ir navegando de una en una en busca de información.

Lo ideal es tener toda la información agrupada en una misma organización, de forma que sea cómodo acceder a ella, actualizarla, consultarla...

Una organización puede contener a su vez varios proyectos, por lo que con una organización para una empresa pequeña puede ser más que suficiente. Por lo general, al momento de registrarnos en Azure DevOps, se crea automáticamente una organización con el nombre del mail que hemos utilizado para este registro.

Para renombrarla, busca en la esquina inferior izquierda el icono de **Organization settings**. Allí encontrarás, en la sección **General**, la información principal de la organización.

The screenshot shows the 'Organization Settings' page in Azure DevOps. The left sidebar contains a navigation menu with categories: Organization Settings, Security, Boards, Pipelines, and Settings. The 'Overview' section is selected. The main content area displays the 'Overview' settings for 'SomeOrganization'. Fields include Name (SomeOrganization), URL (https://dev.azure.com/SomeOrganization/), Privacy URL, Description (Add organization description), Time zone (UTC), and Region (West Europe). A 'Save' button is present, with a note that changes affect all projects and members. At the bottom, there is a 'Delete organization' section with a warning and a 'Delete' button.

Organization Settings
SomeOrganization

General

- Overview
- Projects
- Users
- Billing
- Auditing
- Global notifications
- Usage
- Extensions
- Azure Active Directory

Security

- Policies
- Permissions

Boards

- Process

Pipelines

- Agent pools
- Settings
- Deployment pools
- Parallel jobs

Overview

Name
SomeOrganization

URL
<https://dev.azure.com/SomeOrganization/>
[Learn more about URLs](#)

Privacy URL
[Learn more about the Privacy URL](#)

Description
Add organization description

Time zone
UTC

Region
West Europe
[Learn more about the Region](#)

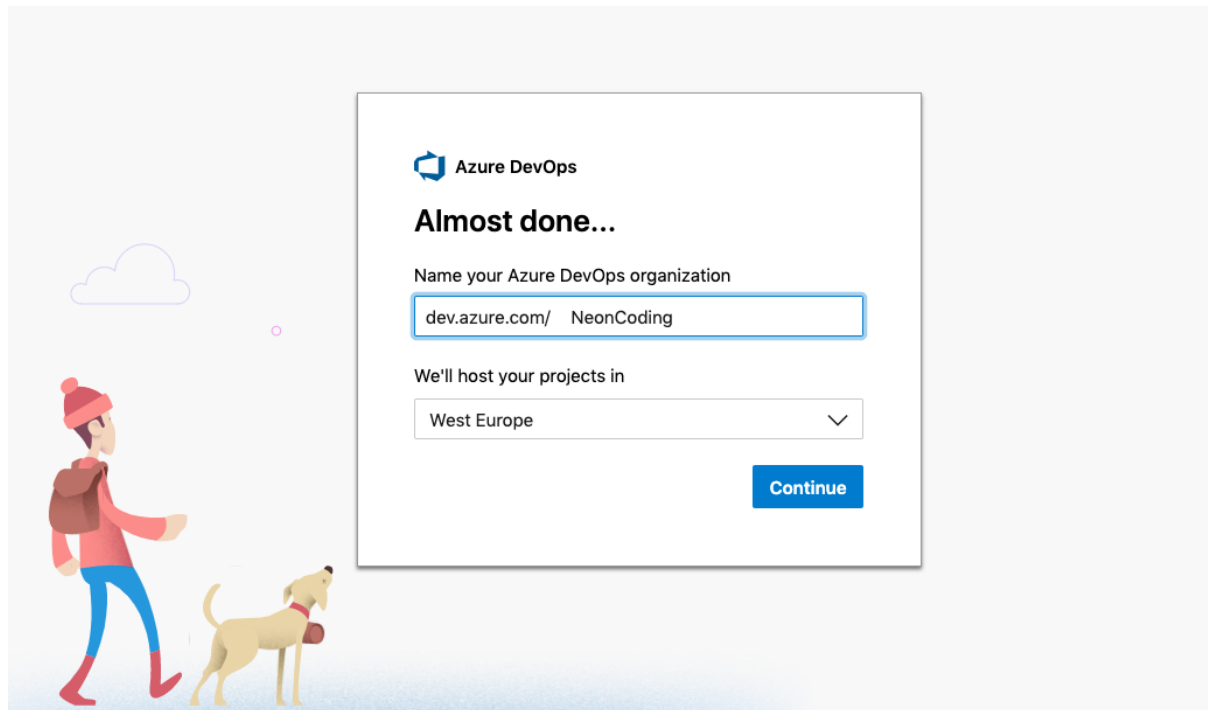
Save ⓘ Changes made will affect all projects and members of the organization

Delete organization
This will affect all contents and members of this organization.
[Learn more about deleting organizations](#)

Delete

Es ahí donde puedes renombrar la organización, añadir una descripción... o si lo necesitas, he aquí una zona de peligro, eliminar la organización, pulsando el botón **De lete**.

Si al entrar en **Azure DevOps** no tenemos ninguna o hemos borrado la que se crea de forma automática, aparecerá la siguiente imagen:



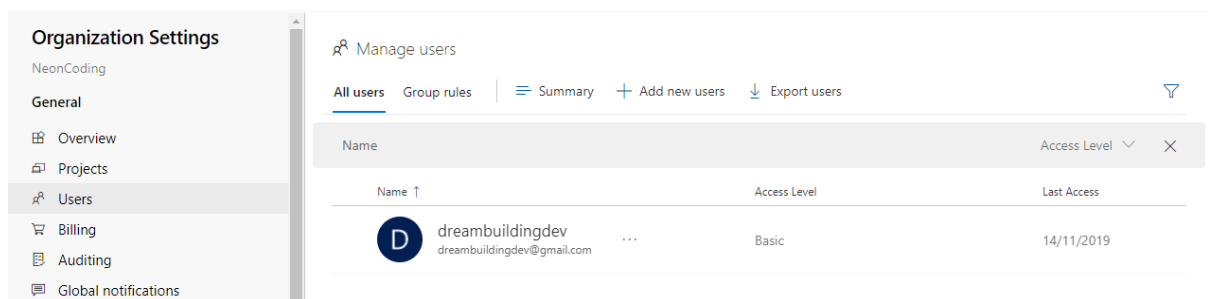
Podemos seleccionar el nombre de la organización, que determinará la URL a través de la que accedemos a ella, así como en qué parte del globo se va a hospedar esta organización. ¡Debe ser un nombre único!

Una vez personalizada al gusto nuestra organización, podemos comenzar a trabajar sobre ella.

Gestión de usuarios

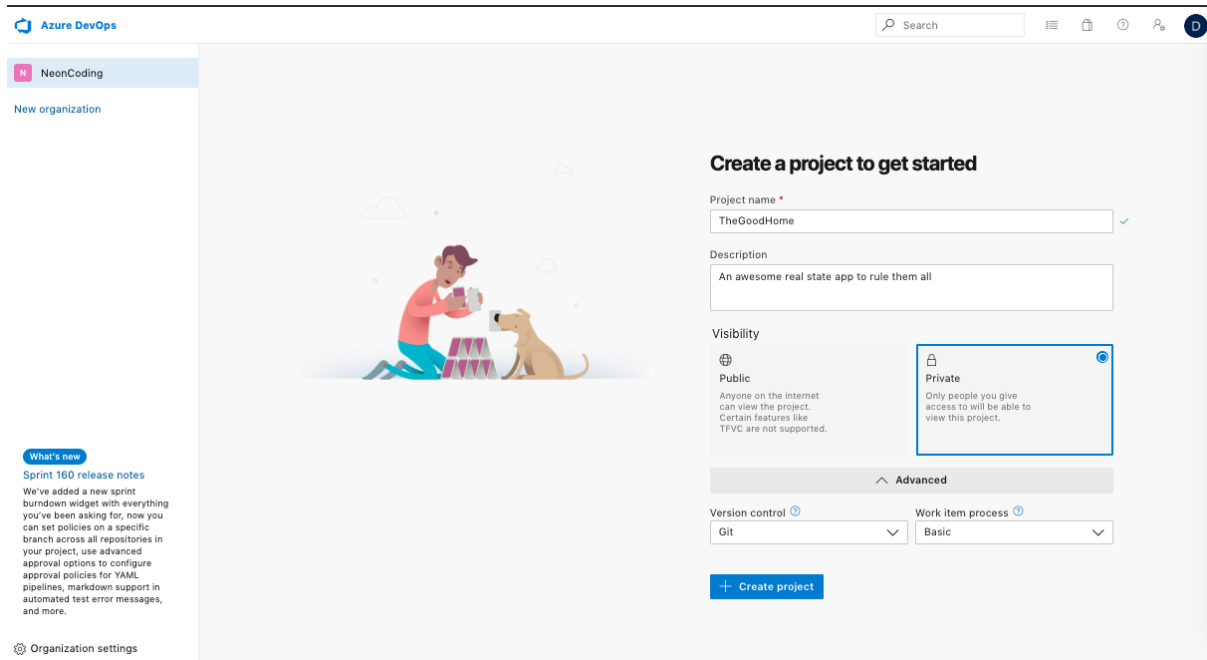
Antes de avanzar, volvamos a la gestión de la organización.

En la sección General, puedes encontrar también la sección de Users. Desde ahí puedes ver y gestionar quién accede a la organización y con qué permisos.



Creación de un proyecto

Comenzaremos entonces creando un nuevo proyecto dentro de nuestra organización.




El equipo de NeonCoding, habiendo creado su organización en Azure DevOps, creará un proyecto para su aplicación de gestión de viviendas y alquileres, "TheGoodHome". Para ello, nombra de forma adecuada el proyecto, le da una descripción, marca su accesibilidad como privada... y, desplegando el botón de Advanced, decide que el sistema de control de versiones será Git y el proceso de gestión de tareas será Basic. Para elegir cual es el adecuado, vamos a sumergirnos en todos los procesos para determinar el más conveniente.

Elección de *work item process*

Existen cuatro tipos de *work item process* o proceso de gestión de tareas: Basic, Agile, SCRUM y CMMI. Todas tienen sus características propias y se ajustan a formas de trabajo o bien quieres adoptar en tu equipo. **En caso de dudas o desconocimiento de los otros, elige el proceso Basic**, es el que veremos usar al equipo NeonCoding.

Tipo	Descripción	Ejemplo
Basic	El modelo más básico, usando Issues, Tasks y Epics para registrar el trabajo.	
Agile	Escoge este proceso si tu equipo usa métodos Agile para planificar, incluyendo SCRUM. Separa a su vez actividades de desarrollo y de testing. Este proceso funciona bien si quieres registrar <i>user stories</i> y (opcionalmente) <i>bugs</i> en el tablero Kanban, o registrar <i>bugs</i> y tareas en el tablero de tareas.	
SCRUM	Si ti equipo practica SCRUM, este es tu modelo. Este proceso funciona muy bien si quieres mantener <i>product backlog items</i> (PBI) y los bugs en un tablero Kanban, o quizás dividir estos PBI y <i>bugs</i> en tareas para el tablero de tareas. Está diseñado para soportar la metodología SCRUM	

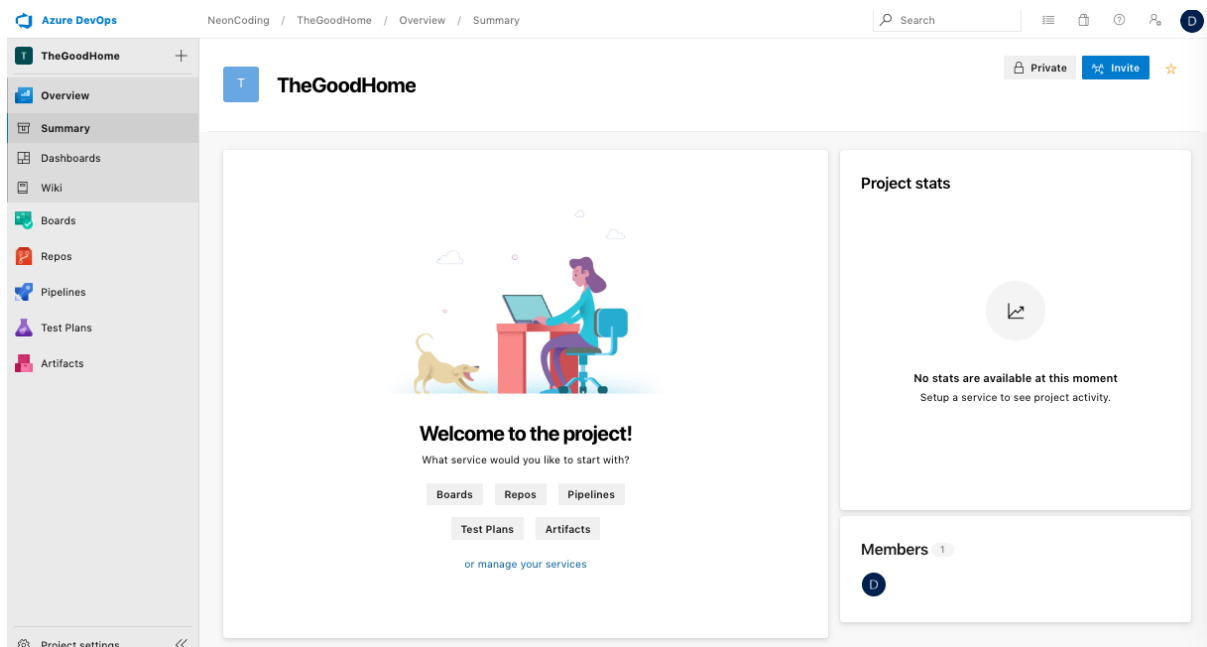
Tipo	Descripción	Ejemplo
CMMI	Si tu equipo sigue métodos de proyecto más formales que requieran un <i>framework</i> para la mejora de proceso y un registro auditable de decisiones, CMMI es la opción. Puedes registrar requisitos, solicitudes de cambio, riesgos, <i>review</i> ...	

Las tareas tienen diferentes estados a medida que las creamos, trabajamos con ellas, las marcamos como finalizadas... y a la vez, cada modelo de los arriba descritos tienen una forma diferente de etiquetar esos estados.

Tracking area	Basic	Agile	Scrum	CMMI
Workflow states	<ul style="list-style-type: none"> To Do Doing Done 	<ul style="list-style-type: none"> New Active Resolved Closed Removed 	<ul style="list-style-type: none"> New Approved Committed Done Removed 	<ul style="list-style-type: none"> Proposed Active Resolved Closed
Product planning	<ul style="list-style-type: none"> Issue 	<ul style="list-style-type: none"> User story Bug (opcional) 	<ul style="list-style-type: none"> Product backlog item Bug (opcional) 	<ul style="list-style-type: none"> Requirement Bug (opcional)
Portfolio backlogs	<ul style="list-style-type: none"> Epic 	<ul style="list-style-type: none"> Epic Feature 	<ul style="list-style-type: none"> Epic Feature 	<ul style="list-style-type: none"> Epic Feature
Task and sprint planning	<ul style="list-style-type: none"> Task 	<ul style="list-style-type: none"> Task Bug (opcional) 	<ul style="list-style-type: none"> Task Bug (opcional) 	<ul style="list-style-type: none"> Task Bug (opcional)

Si quieres seguir explorando, puedes seguir leyendo en la [documentación oficial de Microsoft](#)

Después de debatir, explorar y contemplar posibilidades, NeonCoding crea su proyecto privado con Git, proceso Basic. Deciden sumergirse con tranquilidad en esta aplicación, pues ofrece muchas posibilidades.



Esta es la primera visión de su proyecto, "TheGoodHome", y uno de los primeros pasos hacia el Dev Ops en NeonCoding. Un espacio claro, amplio, y con todo bien organizado para empezar a trabajar.

En el menú lateral a la izquierda se pueden apreciar varias opciones, que son menús desplegables con los que alcanzar diversas tareas.



Tenemos Boards para gestionar nuestras tareas, Repos para almacenar nuestro código con Git...



¿Test Plans? Estoy deseando saber cómo se usa, a ver cómo podemos explotar sus posibilidades para mejorar nuestro _testing_.



Artifacts... parece que ahí estarán nuestro _software_ a entregar, parece interesante cómo se organiza.




Me encantaría saber cómo funcionan las Pipelines, ¿cuando decís que empezamos?

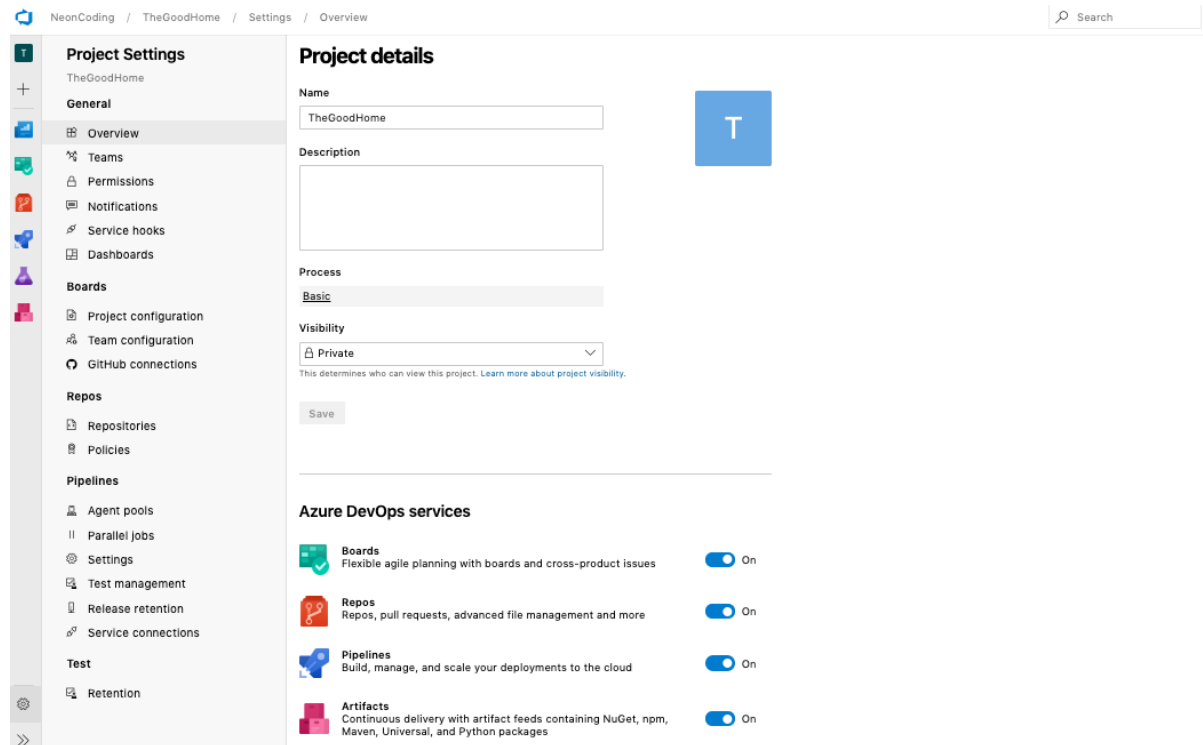
Overview

La primera sección en la cual entramos cuando iniciamos nuestro proyecto, es **Overview**. Es aquí desde donde tenemos una visión global del proyecto. En el menú desplegable, tenemos varias opciones a considerar.

Summary, desde donde podemos ver una visión más generalista de todo el proyecto. Podemos ver las estadísticas del proyecto, las personas que operan en este, así como gestionar los servicios.

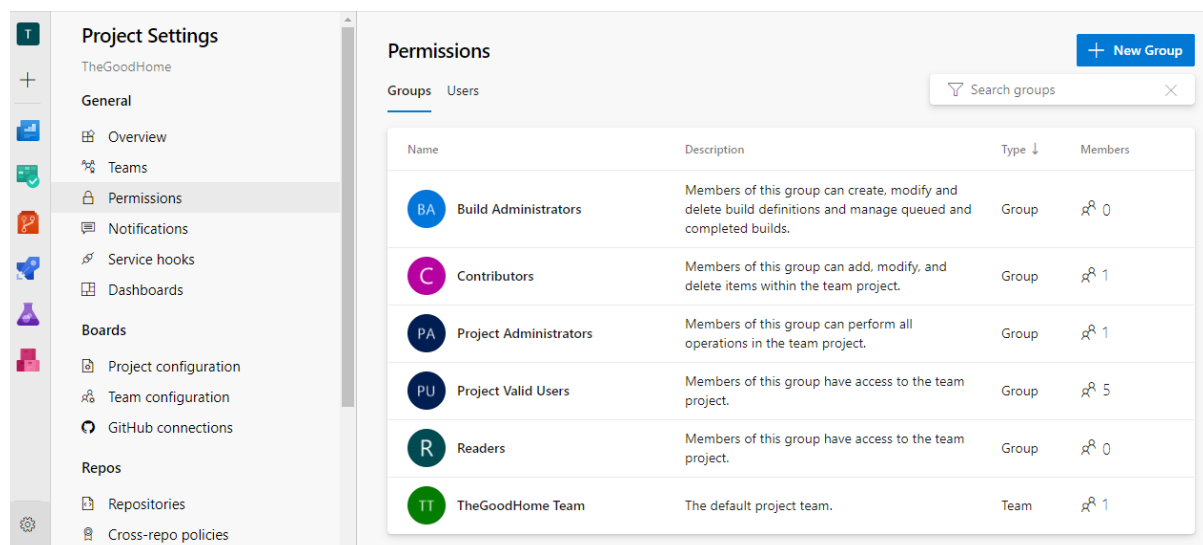
Si observas, tenemos una sección desde la que podemos viajar a todos los servicios disponibles... además de una última opción: **Manage your services**. Es posible que tu proyecto no necesites todos los servicios que tienes disponibles, y quieres eliminarlos de tu vista para centrarte en lo que necesitas. Así que pulsa en esta opción para ver qué contiene.

Además de poder gestionar los detalles de tu proyecto y modificarlos a placer, vemos en la última sección cómo podemos activar o desactivar todos los servicios a los que tenemos acceso. A esta sección también puedes acceder desde **Project settings**, siempre disponible en la esquina inferior izquierda de la aplicación, junto a este símbolo: .



Gestión de usuarios

Si navegamos a la gestión de proyecto, y entramos en **Permissions** podemos ver la gestión de usuarios a nivel de proyecto. Encontramos la pestaña **Groups**, donde se encuentran los diferentes roles en el proyecto.

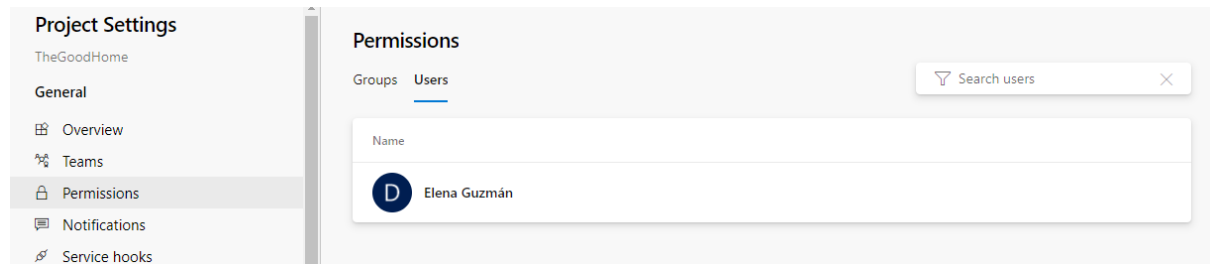


Los roles que encontramos son:

- Build administrators
- Contributors
- Project administrators
- Project valid users
- Readers
- Default project team (con el nombre del proyecto)

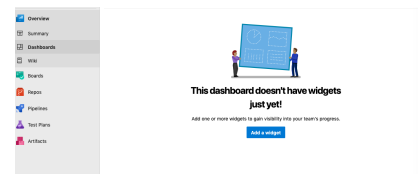
Seleccionando en cada uno de estos roles entraremos en la lista de permisos que podemos otorgar o quitar.

Y en la pestaña de Users, encontramos la lista de usuarios que se encuentran en nuestro proyecto. A medida que invitemos a personas a que se unan, esta lista irá creciendo.

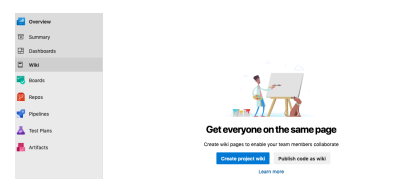


Retrocedemos y regresamos a **Overview** para seguir explorando:

Dashboard, que es un panel donde iremos añadiendo *widgets* a nuestro proyecto. Existen *widgets* para múltiples propósitos y con muchas capacidades para visualizar datos y estadísticas de nuestros proyectos. Es una forma de visualizar nuestro proyecto mucho más completa y por supuesto personalizable.



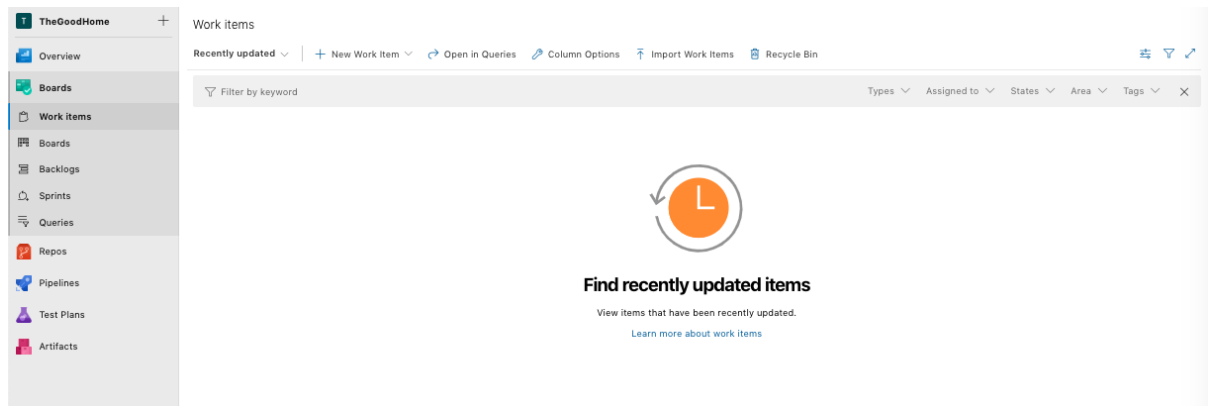
Wiki, para reunir toda la documentación de nuestros proyectos. Debemos crear una nueva wiki para empezar a documentar nuestros proyectos. Las wikis soportan, en su mayoría, Markdown, un lenguaje de marcado sencillo que permite crear documentos formateados de forma muy sencilla. Puedes seguir [leyendo aquí](#).



Boards

Work items

Tal y como Alex señalaba, tenemos como siguiente sección los **Boards**, tableros en los que organizaremos nuestro trabajo. Al estar iniciándonos, no tenemos ninguna tarea. En la barra central tenemos la opción para añadir nuevas tareas, **New Work item**.



Podemos crear tres tipos de elementos:

- Epic
- Issue
- Task

Empezaremos creando Issues, para iniciarnos. Pasaremos a navegar a la página de creación de *work items*, donde especificaremos toda la información necesaria. Las tareas también hacen documentación, necesitan del contexto suficiente como para entender qué se requiere o cuál es el problema a solucionar.

Un Issue necesitará:

- Un título, debe ser descriptivo, es el único elemento obligatorio
- Una persona asignada
- Una estado o *state*
- Una descripción

Todos los *work items*, ya sean Epics, Issues o Tasks, tiene una sección *Discussion*. En caso de necesitar debatir o añadir información adicional al *work item*, este es el lugar. Desde aquí puedes enlazar a *pull requests* o mencionar a personas en el equipo a través de @ seguido por el nombre.

Work Items [Back to Work Items](#)

NEW TASK ● Field 'Title' cannot be empty.

Enter title

Unassigned 0 comments Add tag

Save

Stat: To Do Reason: Added to backlog Area: TheGoodHome Iteration: TheGoodHome

Description

Click to add Description

Discussion

Add a comment. Use # to link a work item, ! to link a pull request, or @ to mention a person.

Planning

Priority: 2

Activity

Remaining Work

Deployment

To track releases associated with this work item, go to Releases and turn on deployment status reporting for Boards in your pipeline's Options menu. Learn more

Development

+ Add link

Related Work

+ Add link

Si creas Tasks también se necesita un título, además de poder añadir descripción, persona asignada, prioridad... Puedes también especificar la *Activity*, para definir de qué trata la tarea entre las siguientes opciones:

- Development
- Design
- Documentation
- Deployment
- Requirements
- Testing

Una vez hayamos añadido toda la información necesaria para el *work item*, pulsamos en el botón Save que tenemos a la derecha. Ahora, si volvemos a *Work items*, veremos un listado con las tareas que hemos creado hasta ahora. Desde ahí puedes gestionarlas de forma sencilla.

Boards

Si navegamos hacia Boards, veremos cómo el proceso Basic determina los estados de nuestros *work items*.

TheGoodHome Team ☆ 🔍

Board Analytics [View as Backlog](#)

Issues ⌵ ⌵ ⌵ ⌵ ⌵

To Do < Doing 0/5 Done

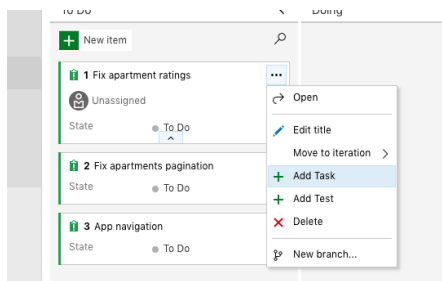
+ New item

1 Fix apartment ratings
State ● To Do

2 Fix apartments pagination
State ● To Do

3 App navigation
State ● To Do

A medida que vayamos actualizando, creando y completando *_work items_*, los iremos moviendo entre los tres tableros para actualizar su estado. Desde Boards también podemos completar nuestros Issues, que están compuestos de Tasks. Si lo

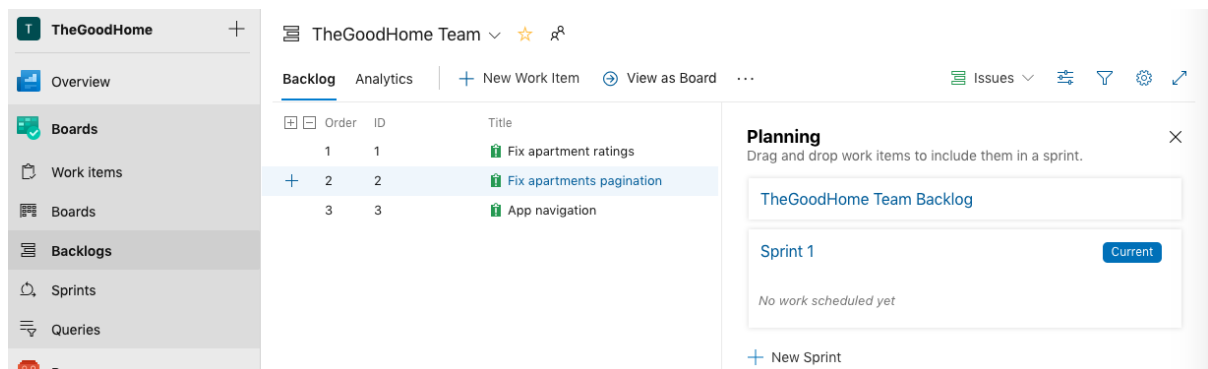


necesitamos, podemos crear nuevas Tasks y marcar si están completadas o no en nuestros Issues

Manejar nuestras tareas y trabajo pendiente es sencillo si lo visualizamos de la forma adecuada y de forma intuitiva.

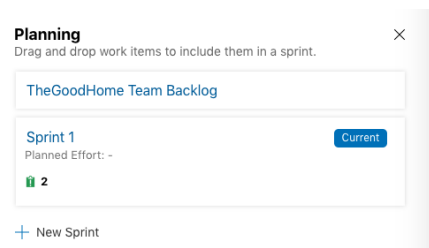
Backlog

Sigamos avanzando, esta vez hacia *Backlog*. El Backlog permite organizar nuestras tareas a través del tiempo. Podemos reordenarlas para marcar su prioridad, asignarlas en *sprints*. Desde aquí también es posible añadir nuevos *work items*, para seguir acumulando trabajo.



En la barra superior también tenemos una opción, llamada **View as board**, que nos permite visualizar nuestros *work items* en un tablero... ¡y precisamente nos hace navegar a Boards!

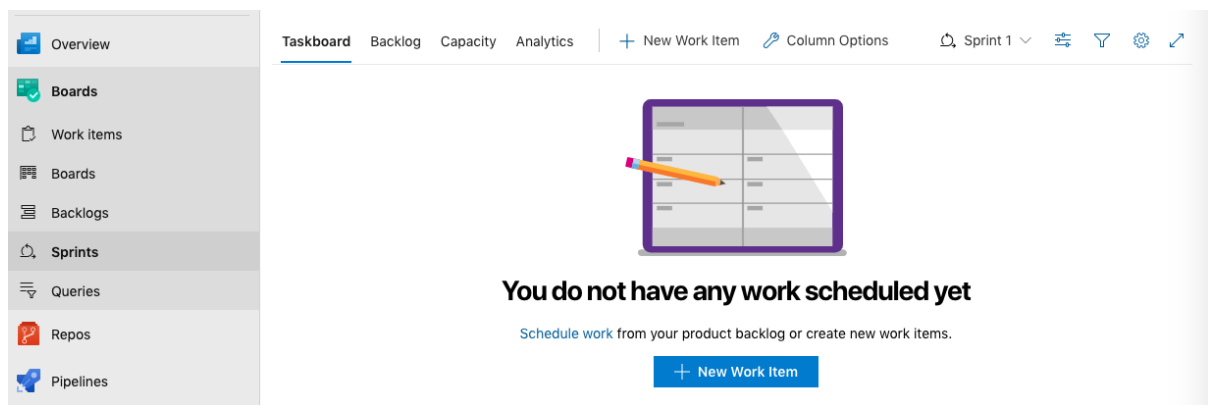
Para añadir un Issue a un *_sprint_*, basta con arrastrar el Issue hacia el Sprint 1, que viene creado por defecto. Cuando hemos añadido un par de tareas, podemos visualizar los *_sprints_* creados y cuantos Issues porta ese *_sprint_* en concreto, como podemos ver en la imagen:



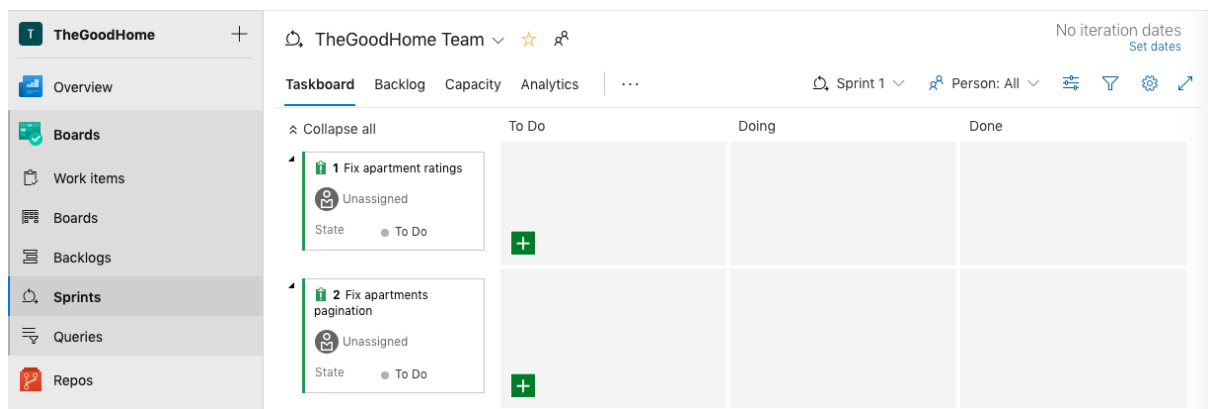
Sprints

En *Sprints*, dividimos en el tiempo las tareas que vamos a acometer. En vez de tenerlas todas a la vez en visualización y sin orden alguno, las dividimos y organizamos para ir resolviendolas poco a poco.

En la imagen podemos observar cómo se visualiza si no hay *work items* en el *sprint*, y por tanto podemos crear un nuevo *work item* y asignarlo de forma directa a ese *sprint*:

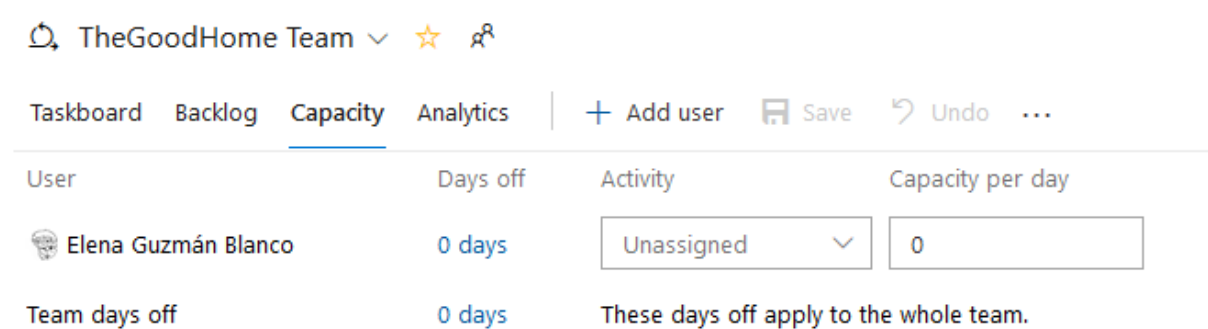


Si el *sprint* que queremos visualizar ya tiene *work items* asignados, se visualiza de la siguiente forma, pudiendo también crear nuevos *items* si fuera necesario.



Si pulsas en la pestaña de Backlog, visitarás de nuevo el listado de *work items* como si estuvieras en la sección Backlog.

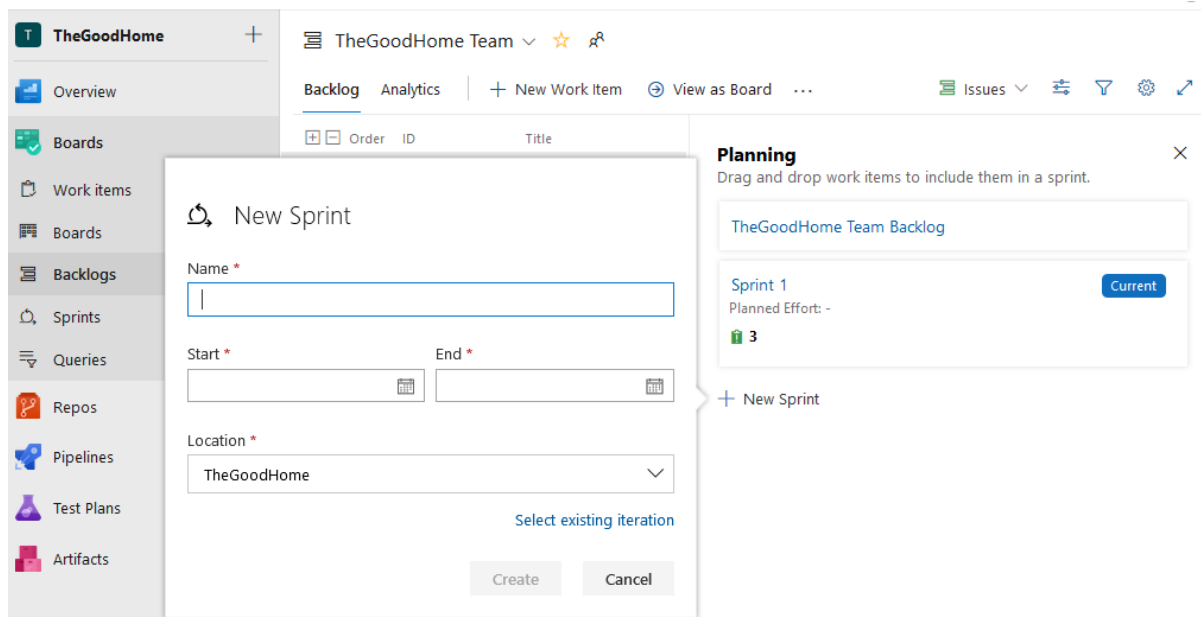
Si pulsas en la pestaña de Capacity, verás la capacidad que tienen todos los miembros del equipo. Las capacidades son las horas diarias que puede dedicar una persona a las tareas de ese sprint.



No todas las personas tienen la misma capacidad. Ya sea por experiencia o por demandas de otros proyectos, puedes especificarlas en esta vista. Y esas horas, ¿a qué se dedican? ¿Diseño, desarrollo, documentación? Es posible especificarlo

Puedes añadir a más usuarios al sprint, si es necesario.

Si el tiempo de un sprint se agota, podemos volver a Backlog y crear un nuevo sprint con un tiempo determinado pulsando el botón New sprint.

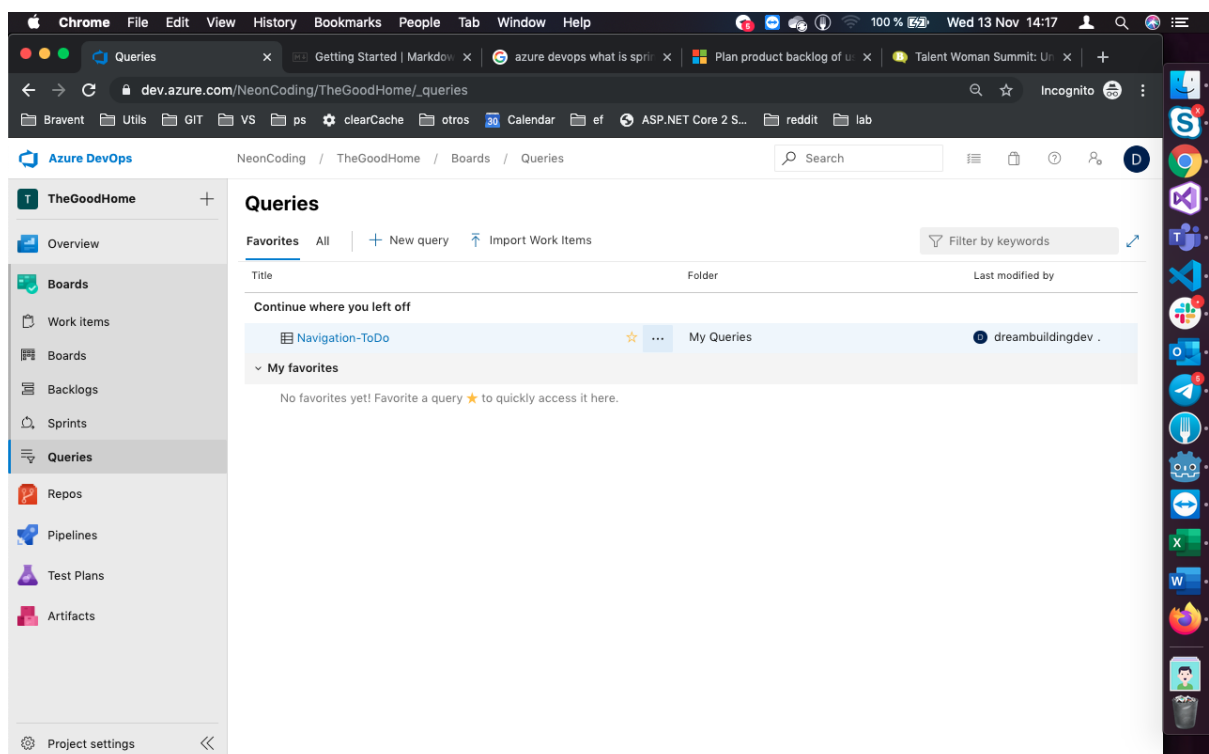


Queries

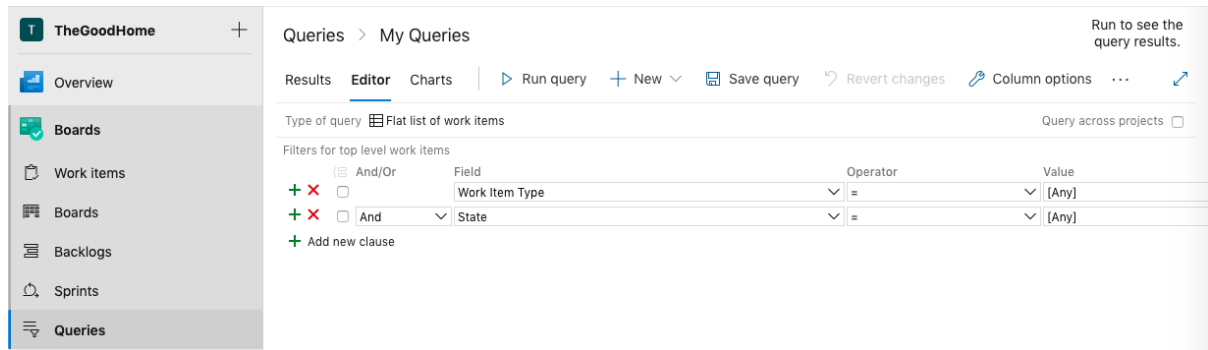
Por último, la sección Queries. Aquí gestionamos consultas, consultas sobre nuestros *work items*. Cuando un proyecto tiene largo tiempo de vida, se han dado de alta muchos *work items* y el trabajo ha fluido sobre Boards de forma constante, es natural necesitar información precisa sobre los elementos con los que trabajamos.

Simplifican el proceso de consultas usando una estructura *SQL-like*. Buscamos por determinados campos que cumplan ciertas condiciones. Pueden existir o no resultados, todo depende de las condiciones que estemos especificando.

Las *queries* que tenemos se pueden almacenar, pues lo más probable es que tengamos que repetirlas varias veces en el tiempo, y eso simplifica el trabajo y minimiza los tiempos. Todas las *queries* que hayan sido guardadas aparecerán en esta lista, que ahora aparece vacía.



Si pulsamos en New query, entraremos en el editor de *queries*, para empezar a crear consultas según nuestras necesidades. Cuando no la hemos editado, aparece de la siguiente forma:

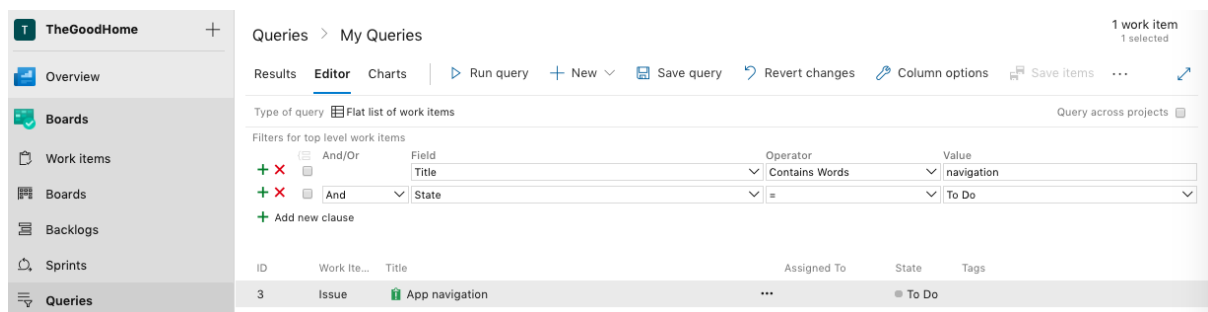


Solo necesitamos encontrarnos con una necesidad que nos haga usar el editor...



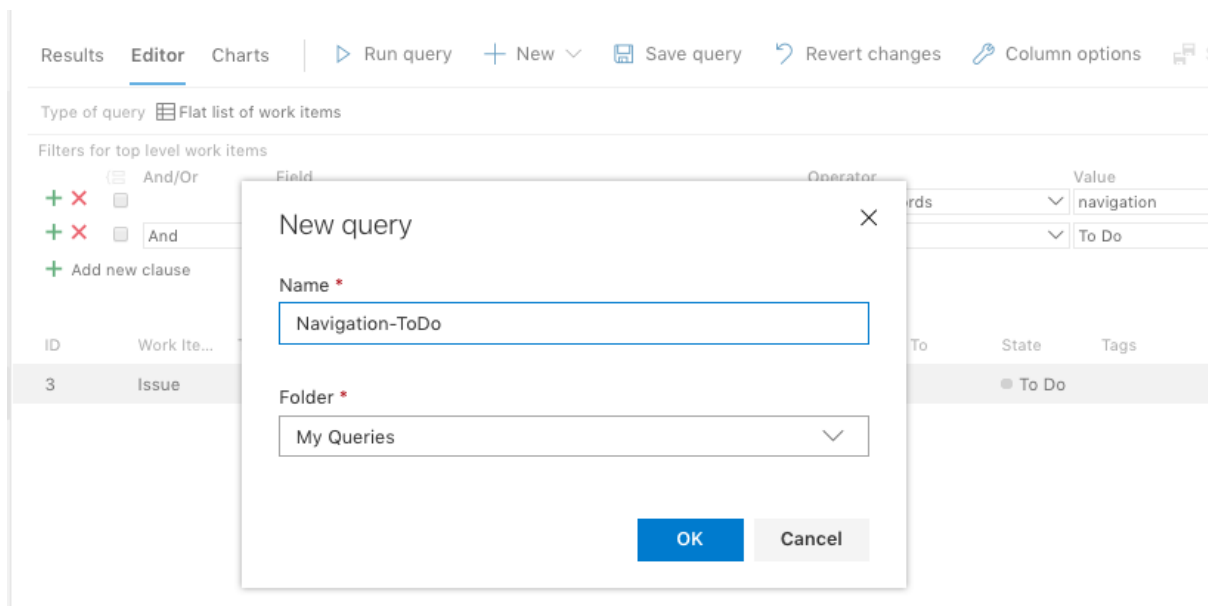
Necesito encontrar todas las tareas que estén por empezar y que tengan relación con "navigation" en su título, me pregunto si podré buscarlas de forma sencilla entre todos los *work items* que tenemos en backlog...

Y la respuesta a Alex es afirmativa, pues usando las Queries podemos consultar de forma fácil esas mismas condiciones que busca. Una vez escrita nuestra consulta, pulsa Run query:

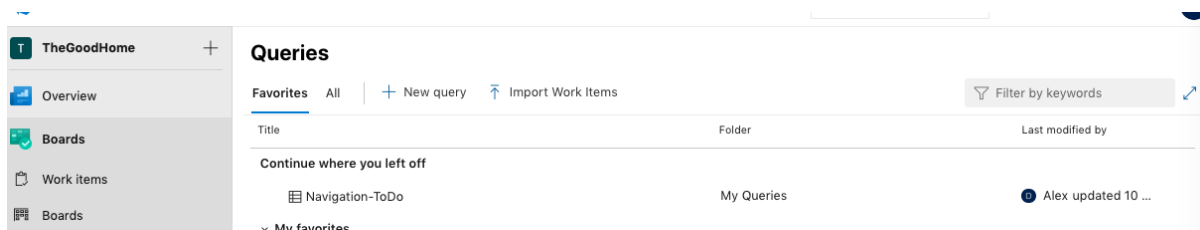


Seleccionando los *Fields* apropiados y determinando los operadores con *Values* acordes, pronto encontramos un resultado para nuestra primera consulta.

Si queremos almacenarla, basta con seleccionar *Save query* para darle un nombre, una carpeta y estará disponible en nuestro directorio de consultas.



Desde aquí podemos gestionarlas con facilidad: elimina, renombra, ejecuta y edita con un par de clicks.



Repos

Llegamos al servicio que almacenará nuestro código, **Repos**. En Repos tenemos varias secciones que nos serán familiares si hemos tratado con otros servicios que gestionan repositorios.

Puedes inicializar un proceso, clonar el repositorio vacío para empezar a programar desde tu equipo o *pushear* un repositorio que ya está vivo hacia



Parece que es hora de subir nuestro repositorio de TheGoodHome a Azure DevOps, ¡manos a la obra!

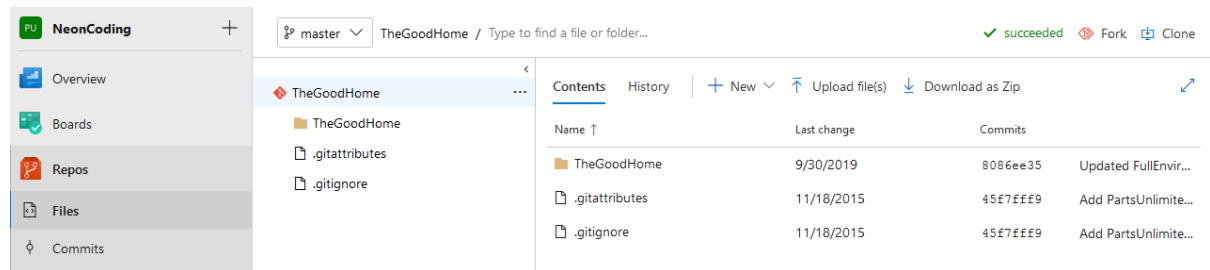
Una vez el equipo está preparado para subir su repositorio a Azure DevOps, puede abrir su terminal para añadir un nuevo *remote* a su repositorio. Este puede tener cualquier nombre, siempre y cuando lo referenciamos de forma adecuada cuando sigamos subiendo cambios.

En el terminal ejecutan:

```
git remote add azdevops https://NeonCoding@dev.azure.com/NeonCoding/TI
git push -u azdevops --all
```

Files

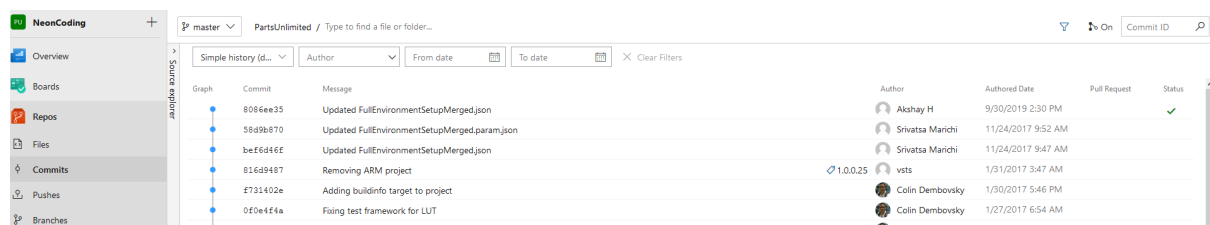
Una vez subido el repositorio, actualizan la página y ya pueden ver todo su repositorio con el código que han estado creando durante todo este tiempo. En **Files**, la primera sección, tenemos acceso a todo el código de nuestro repositorio. Puedes operar con él desde ahí: subir ficheros nuevos, descargar como un *zip*... así como ver todo el historial de *commits*, navegar entre diferentes ramas... ¡todo el código a tu alcance!



Commits

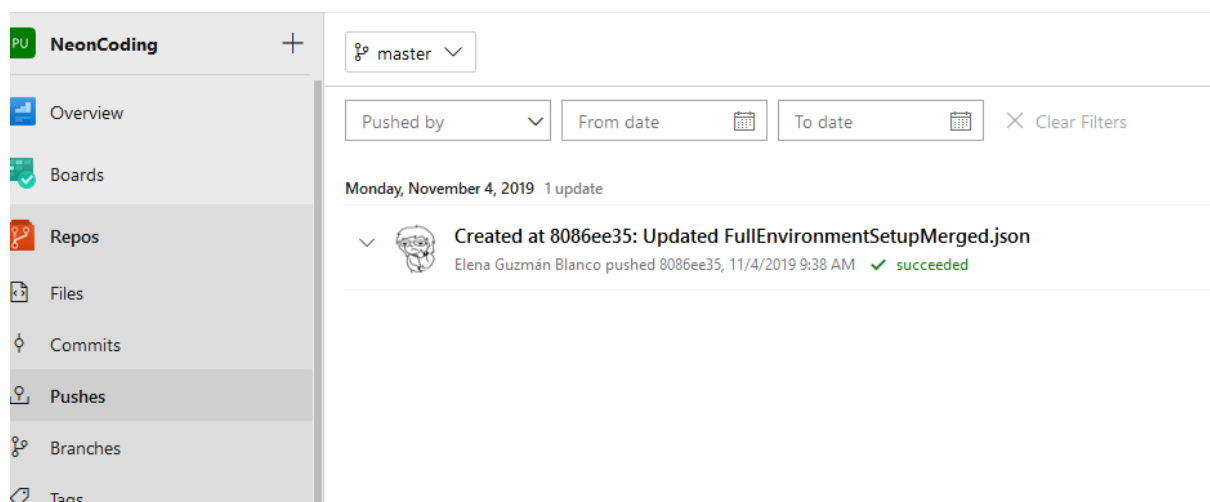
Si necesitas una visión más específica de los commits, operar con ellos de forma más cómoda, tienes la sección *Commits*. Está diseñada para visualizarlos de una forma cómoda y con la que poder ver el cambio en nuestro código.

Permite filtros de autor, fechas, también búsqueda por ID de un *commit*... y contemplar las ramas de nuestros proyectos de la forma más intuitiva.

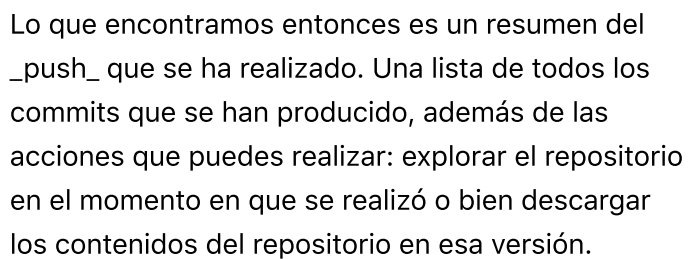


Pushes

En la vista de *Pushes* puedes ver todas las ocasiones en las que el código ha sido subido a un repositorio. Estos *pushes* suelen ser en ramas dedicadas a nuevas características en la aplicación para implementar, y suelen convertirse en *pull requests* en alguna rama.



Si entras en un *push*, verás la siguiente información:

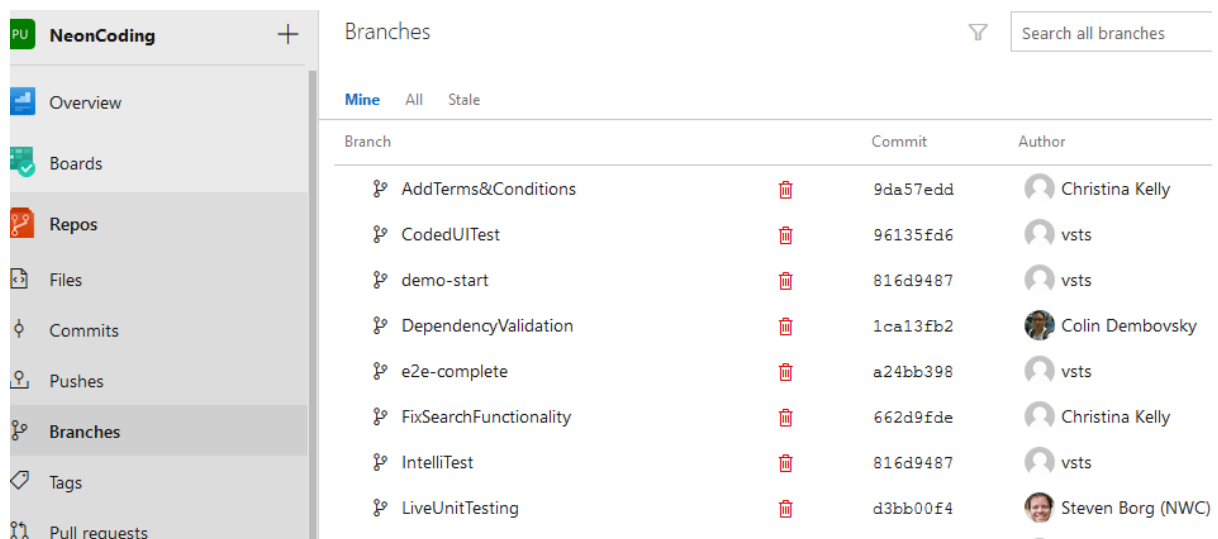


Si navegas a la pestaña de commits, puedes ver con más detalle cada commit realizado desde el inicio del repositorio hasta ese momento.

Branches

Podemos visualizar, igual que toda la información, las ramas que se encuentran en el repositorio.

Dividiendo por las propias y todas, podemos gestionarlas de una forma sencilla. Puedes ver tus nombres, el último *commit*, quien lo ha realizado, el estado, a cuantos *commits* se encuentra por detrás o por delante de la rama principal... o directamente crear un *pull request* a partir de la rama seleccionada.



Las ramas no son eternas, salvo las principales. En algún momento cumplen una fecha de caducidad, pues han cumplido su cometido al contener el código de una determinada funcionalidad. Cuando esta fecha llega, su contenido está desactualizado y ya no tiene más motivos para ser conservada.

Por lo tanto, estas ramas deben ser eliminadas. Pero cuidado: **LAS RAMAS ELIMINADAS NO SE PUEDEN RECUPERAR**. El código que había en ellas **se pierde de forma irreparable**.

Si te encuentras en el caso de que necesitas eliminar ramas para simplificar la visualización de tu repositorio, sigue las siguientes pautas:

- Comprueba si esta rama está activa o es una rama principal, como master o develop
- Comprueba si esta rama te pertenece, la has creado o la has estado utilizando tú
- Si no es tuya, consulta al autor y verifica si puede ser eliminada

Nadie quiere perder código, por eso, establece unas normas, unas políticas de borrado de ramas, como las siguientes:

- Una sola persona debe ser la encargada de borrar las ramas
- Las ramas que vayan a ser borradas deben tener, al menos, un mes de estar inactivas

Establece las normas más adecuadas para tu equipo, y así se evitan riesgos innecesarios de pérdida de ramas y código.

Roles que eliminan branches

Existen determinados roles o grupos que tienen la capacidad para eliminar *branches*, a saber:

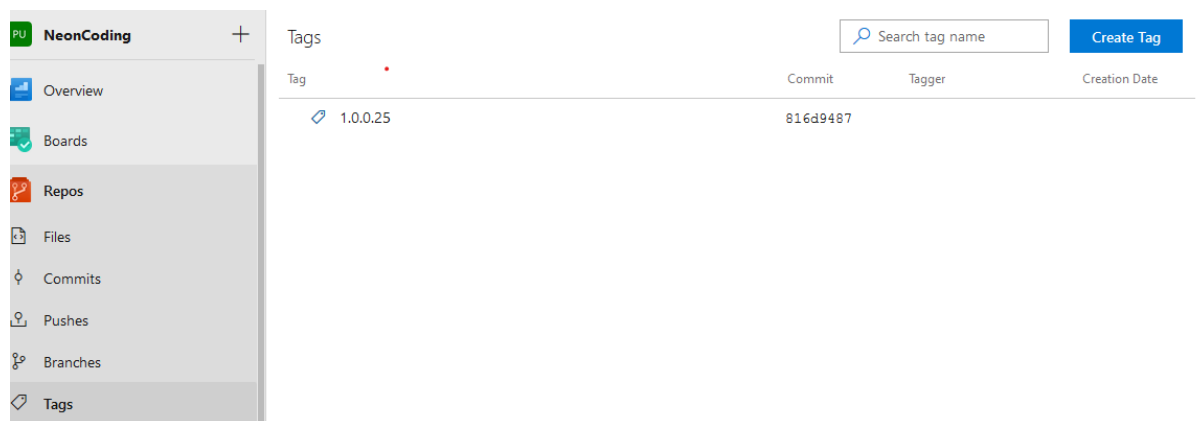
- Admin
- ...

Los grupos de usuarios o roles son editables, como ya sabemos, por lo que podemos modificarlos para proporcionar o arrebatar de esta capacidad a los grupos que creamos necesarios.

Tags

Las *tags* o etiquetas son referencias a un punto específico en el historial de Git. Se captura un punto concreto, un *commit* que se marca para una versión, una *release*. Es como una rama que nunca va a cambiar, se congela en el tiempo pues marca una versión de la aplicación. La aplicación seguirá evolucionando, y llegarán nuevas versiones. Pero una vez se ha creado, no cambiará.

En Tags puedes ver entonces las *tags* que se han realizado sobre la aplicación.



Y si pulsas en el botón **Create tag**, se abrirá un diálogo en el que determinar la información que incluirá la siguiente tag.

El nombre, generalmente un nombre de version, como puede ser **v1.0.11** o **1.2.1...** según la convención elegida por el equipo. Elige también la rama de la cual va a surgir la versión, generalmente la rama principal. Y por añadir más información, una descripción aclaratoria de qué características contiene esta versión que va a ser creada.

Tags

Tag

Commit

Tagger

Creation Date

1.0.0.25

816d9487

Create a tag

Name *

Tag from

🔗 master

✕

Description *

Create

Cancel

Pull requests

Los *pull requests* son las solicitudes de incorporación de código en una rama. Por supuesto, Azure DevOps tiene un sistema para gestionarlo. En esta sección podemos ver los *pull requests* existentes en nuestro repositorio.

Se dividen entre los que has creado y aquellos que se te han asignado para revisar. También, navegando por pestañas, podemos filtrar por aquellos que están activos, completados o están abandonados, las ramas implicadas... ¡incluso búsqueda por el ID del *pull request*!

PU NeonCoding

+

Overview

Boards

Repos

Files

Commits

Pushes

Branches

Tags

Pull requests

Pull Requests

Created by

Assigned to

Target branch

🔍 Pull request ID

New pull request

✕ Clear Filters

Mine

Active

Completed

Abandoned

Created by me

•

Added Terms and Conditions for Sale in Layout.cshtml Page

Elena Guzmán Blanco requested #2 🔗 master

new pull request • 11/4/2019

2

•

Fixed Product Search Functionality

Elena Guzmán Blanco requested #1 🔗 master

Updated 11/4/2019

1

Assigned to me

•

Added Terms and Conditions for Sale in Layout.cshtml Page

Elena Guzmán Blanco requested #2 🔗 master

new pull request • 11/4/2019

2

•

Fixed Product Search Functionality

Elena Guzmán Blanco requested #1 🔗 master

Updated 11/4/2019

1

Si entramos en un Pull Request de los que tenemos citados, veremos cómo se estructura toda la información.

En la primera parte podemos ver su título, su estado, la persona que lo ha realizado y la rama que se quiere mergear y la rama en la que se quiere mergear.

No solo hacemos *pull requests* para mostrar el código, obviamente, debemos revisar el código a conciencia. Por eso debe haber personas que lo revisen, y si cumple con los requisitos, se aprueba. Es por eso que podemos navegar en la descripción, ver los ficheros modificados en *Files*, si el *pull request* se ha ido modificando podemos ver los cambios por cada actualización en *Updates*, el código dividido *_commits* en la pestaña de *Commits*...

The screenshot shows a GitHub Pull Request (PR) interface. At the top, the title is "Added Terms and Conditions for Sale in Layout.cshtml Page" with a status of "ACTIVE". Below the title, it shows the author "Elena Guzmán Blanco" and the branch "AddTerms&Condi... into master". The progress bar indicates "0/2 resolved". On the right, there are buttons for "Approve" and "Complete".

The main content area is divided into two sections. The left section, titled "Description", contains the text "Updated _Layout.cshtml" and "Terms and Conditions for Sale added". Below this is a "Show everything" dropdown and a "Add a comment..." input field. The right section, titled "Work Items", shows a list of work items, including "102 Add Terms and Conditions for Sale ...". Below this is a "Reviewers" section with a list of reviewers, including "Elena Guzmán Blanco". At the bottom, there is a "Labels" section with an "Add label" button.

The bottom section of the PR shows a list of comments. The first comment is from "Elena Guzmán Blanco" dated "11/4/2019" with the text "Updated Terms and Conditions with support clause. Please review." and a status of "Active". The second comment is from "Elena Guzmán Blanco" dated "11/4/2019" with the text "Awesome! Good to go" and two thumbs up emojis. Below the comments is a "Write a reply..." input field and a "Resolve" button.

Esta es toda la información que contiene un *pull request*:






- ID del pull request
- Título
- Descripción de los cambios a incorporar
- Branch objetivo
- Branch origen
- Ficheros modificados
- Commits implicados
- Reviewers
- Work items asociados
- Actualizaciones
- Comentarios y discussion

Es especialmente interesante, en la barra de la derecha, ver a las personas que están de *reviewers* así como los *work items* asociados. Es especialmente importante asociar los cambios incorporados a un *work item*. Nos permite, al momento de completar un *pull request*, marcar ese *work item* como finalizado o Done. Y además navegar hacia las diferentes tareas que vamos completando, pudiendo así acceder a la descripción de la tarea, y también a enlaces desde la wiki... en definitiva, conocer bien el trabajo que se está realizando y cómo se está completando.

Volviendo a la lista de Pull Request, pulsa **New pull request** para comenzar el proceso. Lo primero será seleccionar la rama origen y la rama objetivo.

Una vez seleccionadas estas ramas, se abrirá el formulario para la creación de *pull requests*:

New Pull Request

 CodedUITest  into  master  

Title *

Completing e2e demo

Add label

Description

Completing e2e demo

Markdown supported.

  **B**  *I*       @  # 

Completing e2e demo

Reviewers

Search users and groups to add as reviewers

Work Items

Search work items by ID or title

Create

Introduce un título adecuado, una descripción informativa. Puedes empezar a escribir en **Work items** para buscar aquellos a los que necesites asociar. Funciona de la misma forma con los *reviewers*: busca los nombres de personas en tu equipo para añadirlas como revisores del *pull requests*.

Si haces *scroll down*, podrás revisar todo el código. Cuando todo esté listo, pulsa el botón **Create**. Y un nuevo *pull request* ha sido creado para revisar, denegar o aceptar, ¡que comience la programación!



Tenemos todas las herramientas para gestionar nuestro código al alcance, ¡es genial!



Y además, tenemos control y capacidad para visualizar quién hace qué en todo momento, esto marcha bien...

Los *pull requests* deben ser supervisados por alguien, una mirada que no esté "viciada" del código que se ha escrito y que no ayuden a revisar si el código es de calidad y no esconde errores. Estos ojos serán los *reviewers* en un *pull requests*, y para cada PR puede haber más de un *reviewer*.

Se encargan de encontrar:

- Malas prácticas
- Código de mala calidad

- Problemas de eficiencia

El código que se desarrolla debe ser conocido por todo el equipo, pues la calidad del mismo es responsabilidad de todos.

Pipelines

Antes de introducirnos en Azure Pipelines, hay que entrar en el concepto de *pipelines* en DevOps.

El negocio en esta industria exige entrega continua de valor en el producto, y el valor es creado solo cuando el producto es entregado a un cliente satisfecho. No es cuando un proceso ha sido completado.

La idea clave de todo esto es crear un proceso repetible, confiable y en constante mejora para llevar el software del concepto al cliente. El objetivo es entonces habilitar un flujo constante de cambios en producción a través de una línea de producción de software automatizada.

Ese flujo para crear software de forma automatizado estaría compuesto por diversas tareas: tomar el código, comprobar que compila, pasar todos sus tests... Estas tareas van unas detrás de otras, y debemos poder ejecutarlas todas las veces que necesitemos.

Y eso es lo que conocemos como *pipeline*, el túnel que atraviesa nuestro código, con todas sus fases, para probar su calidad.

La *pipeline* divide el proceso de entrega de software en fases. Cada fase está orientada a verificar la calidad de las nuevas características, validar su funcionalidad y prevenir errores que afecten a los usuarios. La *pipeline* debería proveer de feedback al equipo y visibilidad del proceso de cambios para cualquiera implicado en la entrega de nuevas características.

Una *pipeline* de entrega, o *delivery pipeline* habilita el flujo de pequeños cambios de forma más frecuente. Así el equipo se puede concentrar en optimizar la entrega de cambios que aporta valor cuantificable al producto. Este enfoque lleva a que los equipos de desarrollo monitoreen de forma constante y aprendan dónde encuentran obstáculos, resolviéndolos y gradualmente mejorando el flujo de la *pipeline*. A medida que el proceso avanza, el ciclo de feedback provee de más datos sobre los problemas y obstáculos que deben ser resueltos.

Una *pipeline* es la clave de un ciclo de mejora continua.

Pero antes de avanzar y seguir aprendiendo sobre *pipelines*, es necesario consultar otros conceptos importantes relacionados con este servicio:

¿Qué es una *build*?

Una *build* es una compilación, el acto de compilar el código que hemos creado y todas sus dependencias, para obtener un fichero ejecutable que podemos ejecutar en cualquier entorno: en nuestro equipo de pruebas, en el servidor de producción...

Para que la *build* tenga éxito el código debe compilar, pasar todos los tests que se hayan programado y cumplir con los mínimos de calidad que se hayan impuestos en el equipo.

¿Qué es una *release*?

Una *release* es la compilación de una versión de la aplicación, que va a ser entregada en círculos de usuarios públicos o privados. En metodologías ágiles, es un paquete de software a distribuir después de varias iteraciones de desarrollo y mejora del producto.

En definitiva, es la versión del software que se distribuye.

¿Qué es un *deployment*?

Un *deployment* o despliegue es el acto de hacer accesible el software que estamos creando al usuario, ofrecer una *release* o unidad entregable de software a los usuarios que van a usarlo.

Es el acto de tomar un ejecutable y instalarlo en el equipo o servidor donde va a ser usado por los usuarios finales, que es el objetivo de DevOps: la entrega de valor.

¿Qué es un *environment*?

Un entorno, *environment* o *stage* como también es llamado es un sistema informático en el cual una aplicación o componente de software es desplegado y ejecutado.

Para alcanzar la mayor calidad en el software y no afectar a características ya implementadas con otras que se están mejorando o desarrollando, es común tener múltiples entornos. La configuración más común es tener estos entornos: entorno de desarrollo, donde se ejecutan los nuevos cambios y características; entorno de *testing*, para realizar pruebas de todo tipo; entorno de producción, que es el que el cliente final, los usuarios finales, acaban usando y disfrutando.

Esto permite no mezclar datos, feedback y resultados de las diversas fases de desarrollo.

En futuras unidades veremos con más detalle estos conceptos y el modo en que resultan clave para optimizar nuestro desarrollo bajo las prácticas DevOps. Volvamos entonces con *pipelines*, ahora que conocemos más información y conceptos útiles.

Una *pipeline* se compone de las siguientes fases, que consisten en su mayor parte en la automatización de procesos:

- Build automation y Continuous Integration
- Test automation
- Deployment automation

Build automation y Continuous Integration

La *pipeline* empieza a compilar para crear entregables que pasarán por el resto de fases. Las nuevas características serán implementadas por el equipo de desarrollo e integradas en la base central de código en una base continua, para ser compilada y ejecutada por tests unitarios. Este es el feedback más directo que vamos a obtener, directamente informa a l equipo sobre la salud del código.

Test automation

En esta fase, la nueva versión de una aplicación es testeada de forma rigurosa para asegurar que cumple todos los mínimos de calidad. Es importante que todos los aspectos relevantes (funcionalidad, seguridad, *performance* y *compliance*) son verificados por la *pipeline*. Puede implicar diferentes tipos de tests automatizados o manuales (al menos en inicio).

Deployment automation

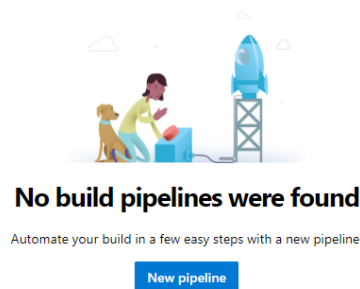
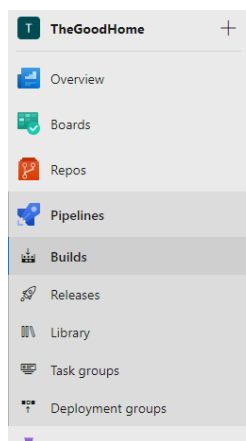
Un *deployment* o despliegue se requiere cada vez que la aplicación es instalada en un entorno (o *environment*) para testing, . Las fases anteriores han verificado la calidad general del sistema, pero estas pruebas son de bajo riesgo. El despliegue puede realizarse en diferentes entornos, con una nueva versión que sea instalada en un subconjunto de entornos de producción, para monitorizarlo antes de ser lanzados por completo al cliente.

El despliegue es automatizado, permitiendo entregas confiables de nuevas funcionalidades a los usuarios en cuestión de minutos, si es necesario.

Todos estos conceptos, toda esta estructura de automatización de nuestras aplicaciones desde su concepción hasta la entrega de cliente, puede ser soportada y gestionada a través de Azure DevOps.

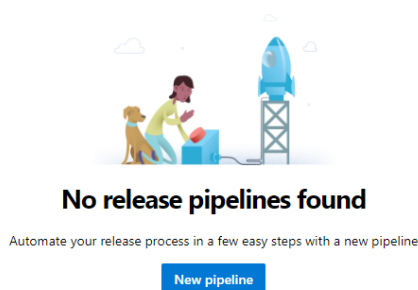
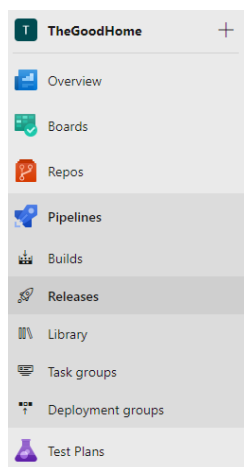
Builds

A través de esta sección, *Builds* creamos nuestras *pipelines* de compilación. Las *pipelines* son procesos, procesos compuestos de tareas o *tasks*. Nuestra labor en estas *pipelines* es especificar estas *tasks*, ya sea en forma de *script* o a través de una interfaz gráfica como la que nos ofrece Azure DevOps, para compilar, testear y desplegar si es necesario. Próximamente crearemos nuestra propia *build pipeline* para ver todo su potencial.



Releases

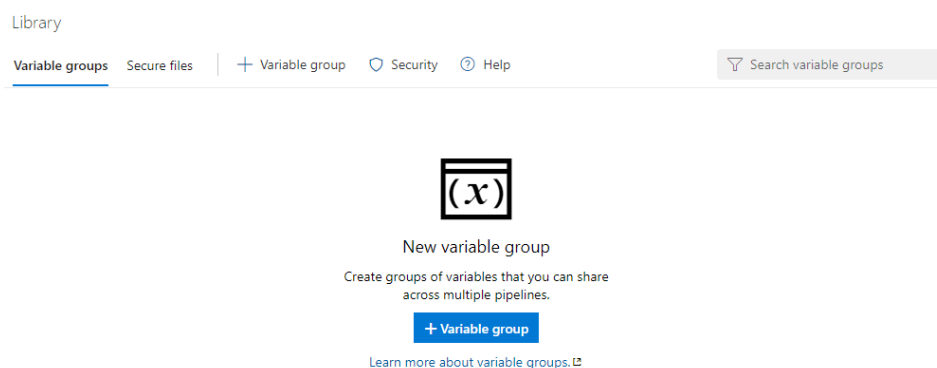
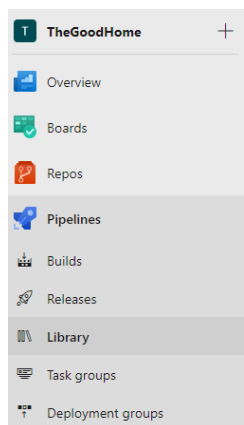
A diferencia de *Builds*, *Releases* nos permite crear *pipelines* para hacer *releases* para los diferentes entornos que tengamos en nuestro desarrollo.



Library

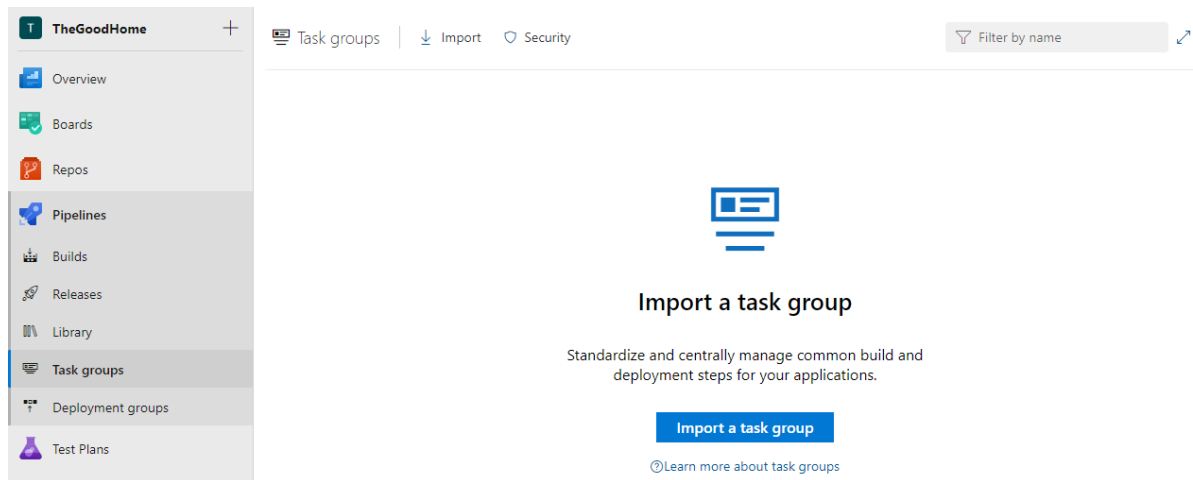
Nuestras *pipelines* están compuestas de *tasks*, y estas tienen que tomar valores concretos para realizar la tarea encomendada. ¿Qué versión de .NET debo utilizar para mi *pipeline*? ¿En qué directorio se encuentran los *tests* que debo ejecutar?

Esta información, que probablemente necesites reutilizar de una *pipeline* a otra, son variables, variables que puedes almacenar para su reutilización. Y es en Library donde se almacenan estas.



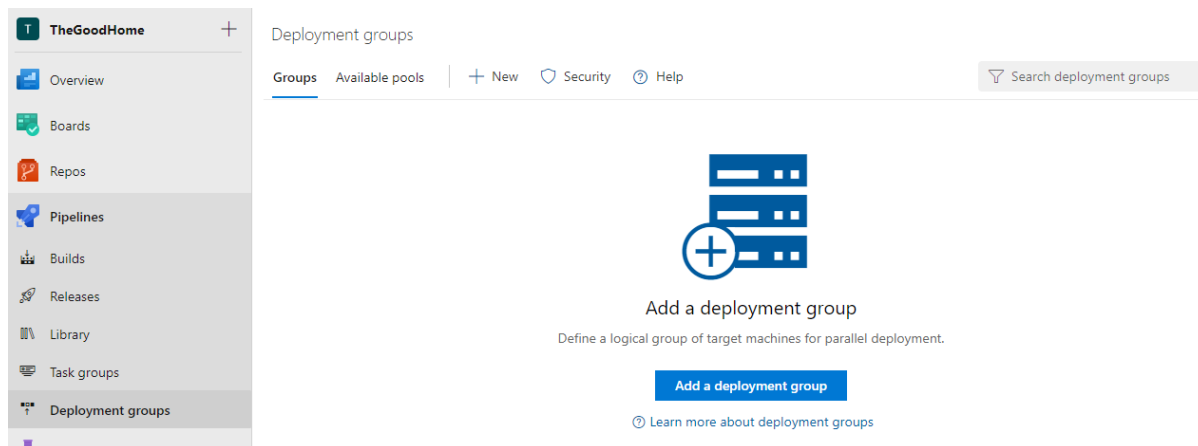
Task groups

Gran parte de la cultura DevOps es la reutilización de materiales, para optimizar tiempos y no dedicar esfuerzos a tareas repetitivas que pueden ser gestionadas por una máquina. De la misma forma, podemos almacenar *tasks* que se ejecutan en conjunto a menudo en Task groups.



Deployment groups

Un grupo de despliegue o *deployment group* es un conjunto de máquinas objetivo para despliegues. Representan entornos físicos, máquinas reales en las que realizaríamos nuestros despliegues.



Artifacts

Este servicio de Azure DevOps está íntimamente relacionado con lo que conocemos con la gestión de dependencias, es por eso que nos introduciremos primero en sus principales conceptos.

En el mundo del software, pronto podemos ver cómo nuestras aplicaciones están compuestas por fragmentos de código que, en conjunto, acaban siendo una aplicación completa con muchas funcionalidades. En ocasiones, estos fragmentos de código no pertenecen a la aplicación en la que estamos trabajando, a veces tampoco pertenece al equipo de desarrollo con el que trabajamos, si no que los tomamos de fuentes externas.

Este código de fuentes externas es lo que llamamos una dependencia, código ajeno a nuestra aplicación que podemos usar con el consentimiento de sus creadores. Este código debe estar bien testeado para evitar fallos, analizado para saber si se ajusta a nuestras necesidades, si podemos cumplir con los términos para usar el código...

Packages

Un package es un modo formalizado de crear una unidad de software y distribuirla, que puedan ser consumidos por otras soluciones o aplicaciones.

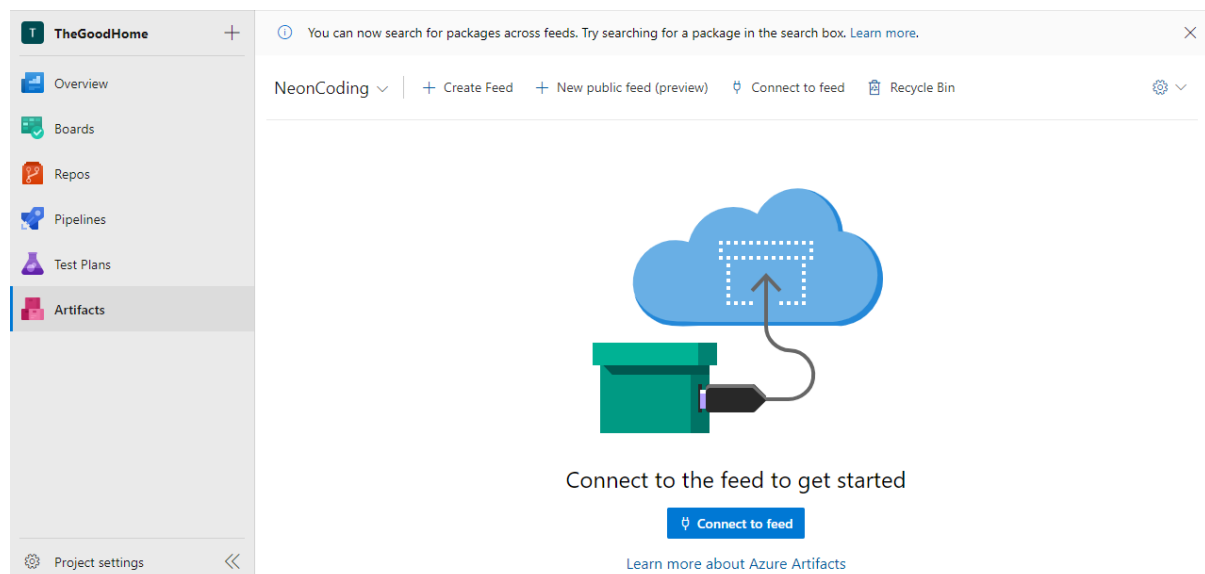
Los packages evolucionan y tienen diferentes versiones con sus diferentes funcionalidades.

Los packages tienen versiones. Las funcionalidades cambian a lo largo del tiempo, y es por eso que necesitamos diferenciar bien los packages que tienen unas funcionalidades y otras que se han ido añadiendo con el tiempo.

Package feeds

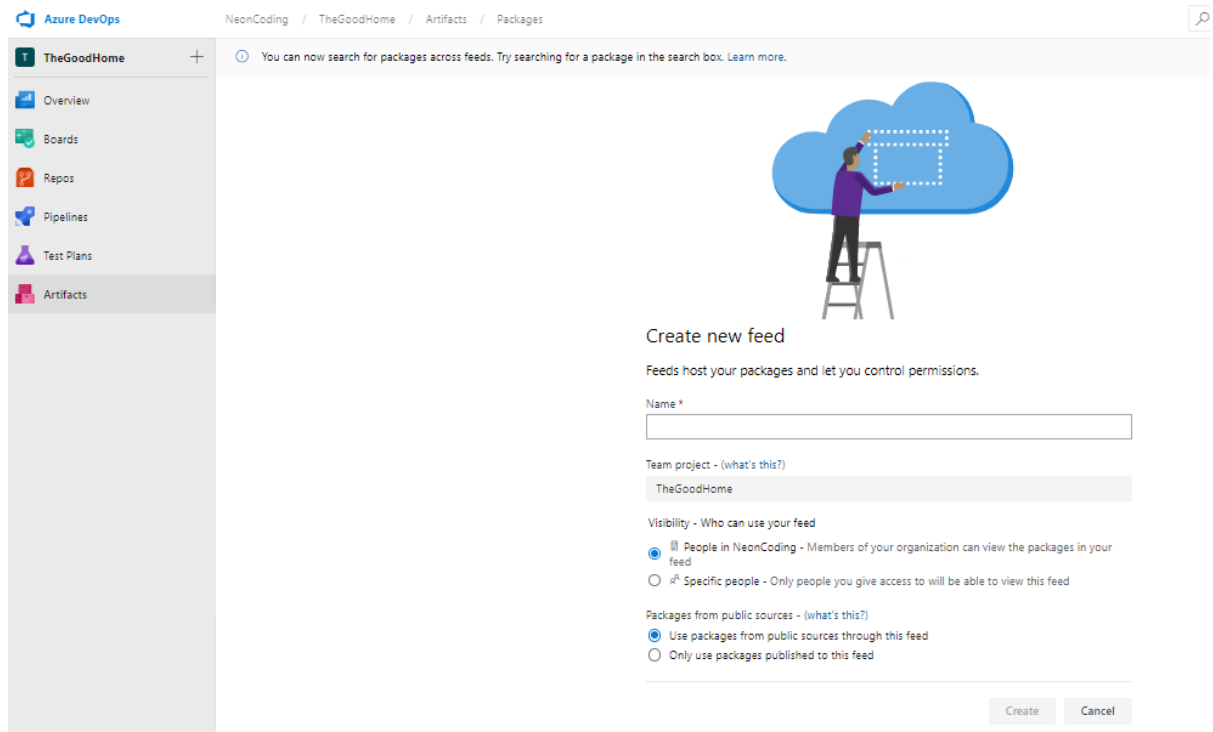
Los *packages* deberían ser almacenados en un lugar centralizado para su distribución y consumición, facilitando así que equipos de desarrollo puedan acceder a las dependencias que esos componentes ofrecen.

Estos almacenes son comunmente llamados *package feeds*, y esto es precisamente lo que podemos gestionar en Artifacts.



Si pulsamos en el botón **Create Feed**, pasaremos a crear un *package feed* en el que almacenaremos nuestros *packages*. Así, todas las personas que tengan acceso a él, podrán acceder a ellos e incorporarlos en sus proyectos y aplicaciones.

A medida que desarrollemos y evolucionemos nuestros *packages*, iremos lanzando nuevas versiones de nuestros *packages*. Porque recuerda: **una versión no debe ser modificada**. Si necesitas actualizarla o mejorarla, desarrolla para lanzar una nueva versión. Pero las modificaciones sobre una versión de *package* ya liberada puede causar grandes problemas de compatibilidad para quienes la están usando.



En futuros módulos crearemos un *package feed* y lo utilizaremos para insertar dependencias en nuestros proyectos, viendo así su utilidad y lo sencillo que es actualizarlos conforme estas dependencias avanzan.

Test Plans

El *testing* es quizás una de las tareas más complicadas y a la vez más beneficiosas en el mundo del software. Es por esto que en Azure DevOps encontramos este servicio únicamente dedicado al *testing* de aplicaciones.

El *testing* nunca se acaba, pero podemos hacer el suficiente como para garantizar que la aplicación funciona de la forma correcta.

Existen muchos tipos de tests:

- unitarios, que se realizan en las *pipelines* de compilación y miden si las funcionalidades añadidas funcionan como se espera.
- integración, donde nuestra aplicación se ejecuta en un entorno donde podemos encontrar fallos no esperados en el primer tipo.
- exploratorios, que miden los flujos de uso en la aplicación.
- de carga, que comprueban que se soportan peticiones y acciones de los usuarios en masa sobre la aplicación.

Test Plans

El *testing* es quizás una de las tareas más complicadas y a la vez más beneficiosas en el mundo del software. Es por esto que en Azure DevOps encontramos este servicio únicamente dedicado al *testing* de aplicaciones.

Test plans

Los test plans nos permiten probar nuestra aplicación a través del uso.

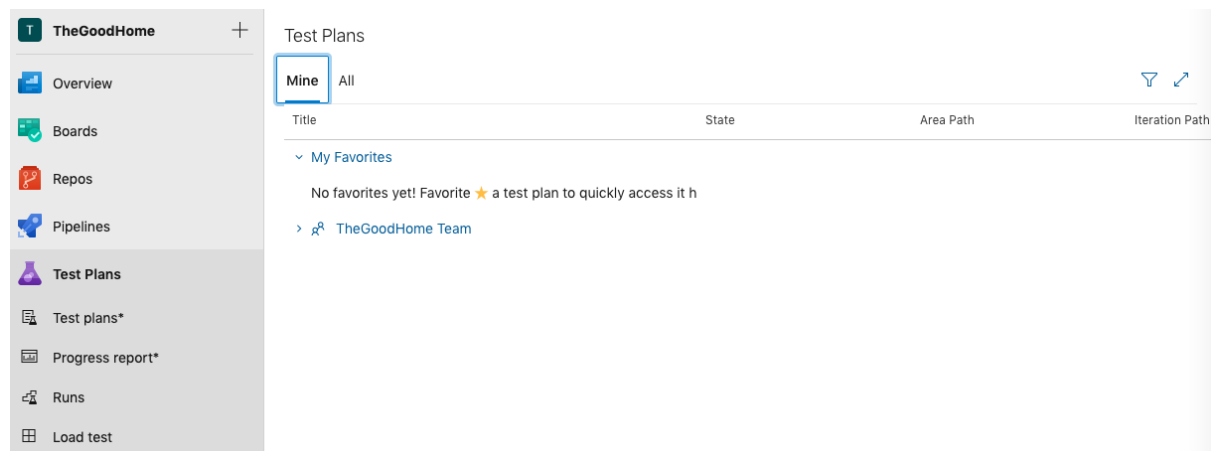
Es lo que llamamos *tests exploratorios*. Indican un flujo a seguir y en el que debemos encontrar un flujo específico.

Por ejemplo: entramos en una aplicación web, y mientras navegamos por ella, introducimos un *input* (un click a un botón, navegar a una pestaña, ordenar una tabla por un determinado campo) esperando un *output* (abrir una notificación en un *popup*, obtener una lista de usuarios registrados, una ordenación correcta en la tabla).

Abrimos una web, abrimos el listado de usuarios, filtramos por un término específico, y esta búsqueda debe resultar en un resultado específico, es decir, los usuarios que encajen en ese filtro.

Los *test plans* nos permiten crear tests a partir de pasos a seguir en nuestra aplicación, pasando en cada uno de esos pasos comprobando si es posible cumplirlos y si el resultado final es el esperado.

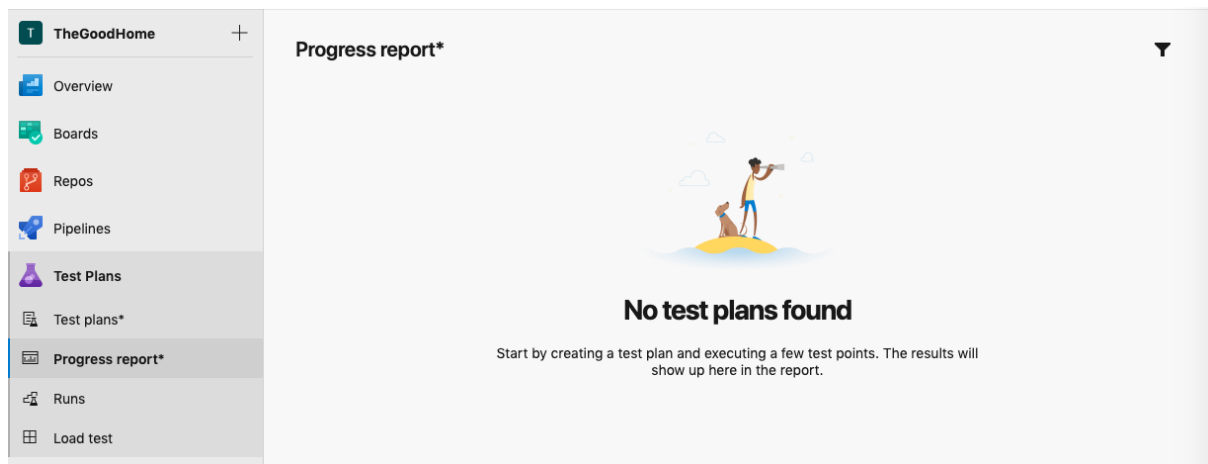
En caso de no cumplirse, es posible crear un *bug*, un *work item* en nuestro Board para tener constancia de una funcionalidad que no es correcta y no está dando el resultado esperado. A partir de ahí, el flujo de mejora continua de nuestra aplicación sigue su curso.



Progress report

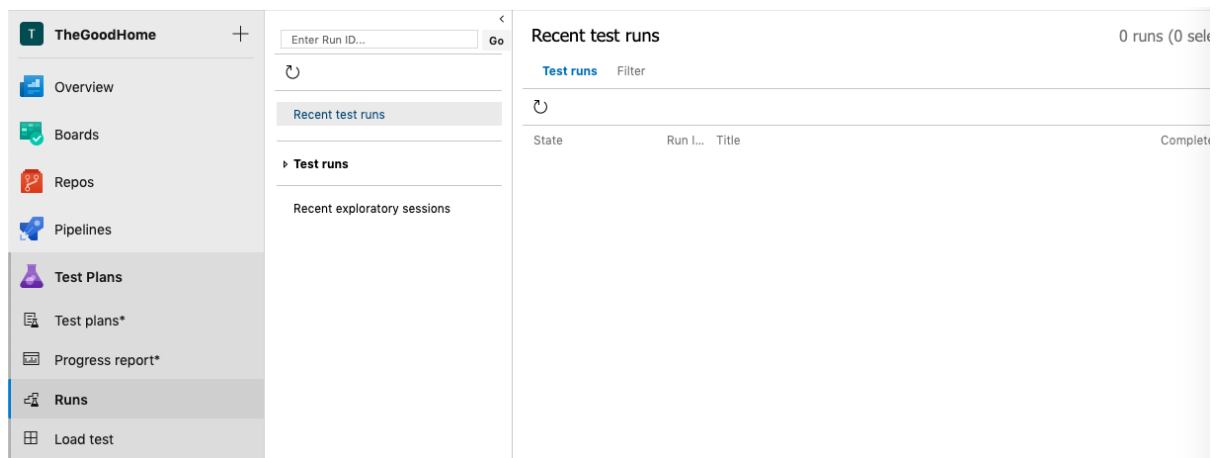
Vamos cumpliendo y realizando *tests plans* a lo largo del tiempo, y en esta sección podemos ver todos los *tests plans* que hemos realizado, los que quedan por resolver y los resultados de todos ellos.

También las importantes cifras a partir de estos: porcentajes de éxito, de *tests* ejecutados... para evaluar el trabajo realizado y el que queda por realizar.



Runs

Tras la ejecución de nuestros tests, necesitaremos ver los resultados de los mismos. Esto es posible en la sección de Runs.



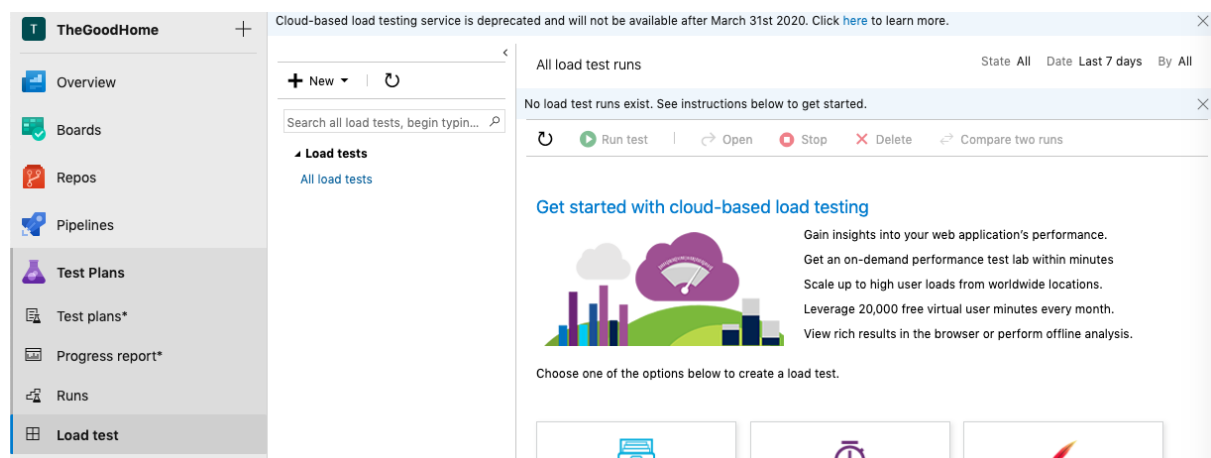
Una vez se ejecutan los tests, podremos ver el informe que este genera. ¿Cuántos tests han tenido éxito? ¿Cuántos han fallado? ¿Hay mensajes de errores que se hayan producido durante la ejecución? ¿Quién ha ejecutado el test?

Contamos incluso con una herramienta de búsqueda para los tests ejecutados basada en consultas, como las *Queries* de nuestros *work items*.

Load tests

Los *load tests* o tests de carga son aquellos que ponen a prueba la capacidad de una aplicación para soportar el flujo y actividad de los usuarios. Se trata de simular las demandas de las diferentes interacciones de los usuarios y medir las respuestas por parte de la aplicación.

Son quizás los tests más invisibles, en el sentido en que tendemos a ser optimistas con respecto a la actividad de nuestros usuarios.



Desde esta vista podemos visualizar toda la información que los tests de carga ofrecen, para así actuar en consecuencia y mejorar nuestro desarrollo.