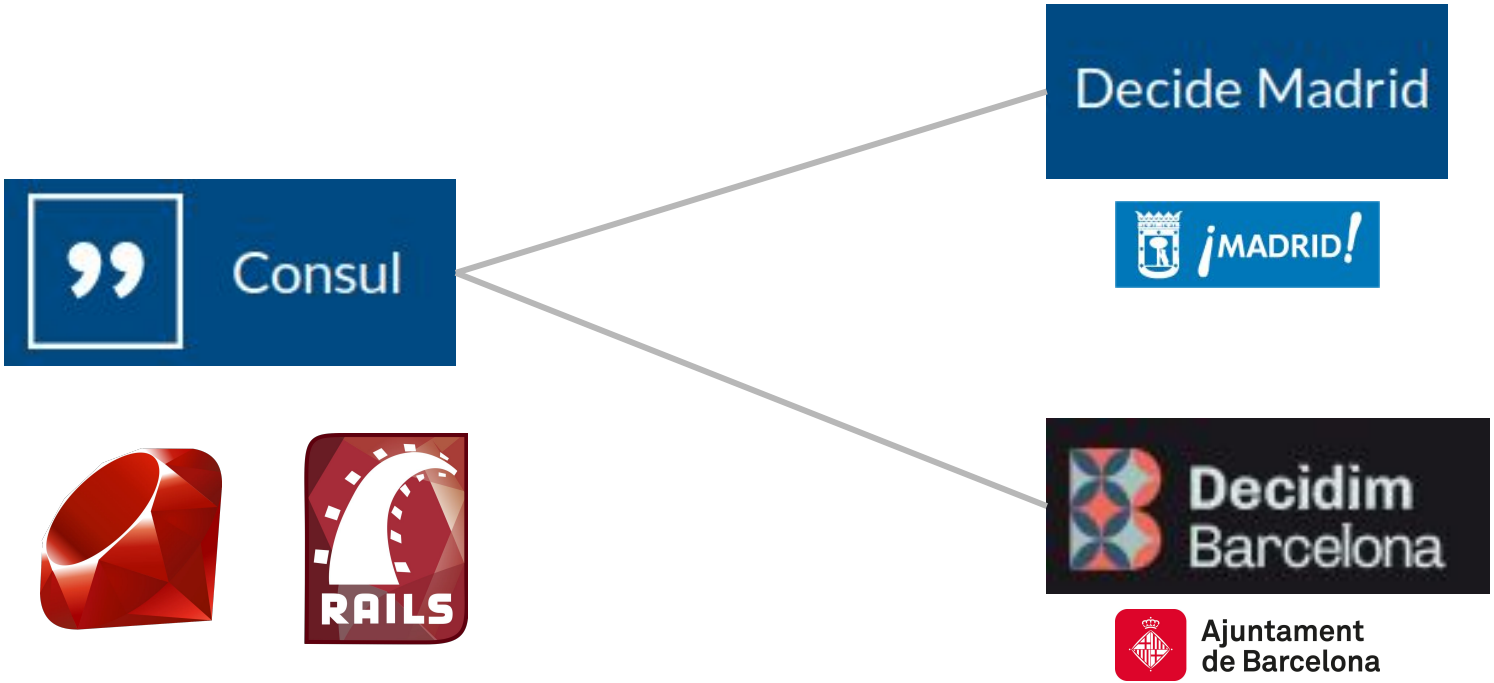
Arquitecturas Ruby On Rails

Cómo facilitar la escalabilidad y colaboración

aLabs*

Ruby on Rails, Consul y Decidim



Extensión de Consul

A PORTA ABERTA

Ayuntamiento de A Coruña
Concello da Coruña

Consul

Decide Madrid

¡MADRID!

participación información transparencia
OVIEDO.es
AYUNTAMIENTO

Decidim Barcelona

Ajuntament de Barcelona

decidimVLC
CONSULTA CIUTADANA D'INVERSIONS EN BARRIS
AJUNTAMENT DE VALÈNCIA

Extensión de Consul

Nombre del fork	Mantenedor	URL	URL Github
Consul	Ayuntamiento de Madrid	No aplica	https://github.com/consul/consul
Decide Madrid	Ayuntamiento de Madrid	https://decide.madrid.es/	https://github.com/AyuntamientoMadrid/consul
decidim.barcelona	Ajuntament de Barcelona	https://decidim.barcelona/	https://github.com/AjuntamentdeBarcelona/decidim.barcelona
Consulta Oviedo	Ayuntamiento de Oviedo	http://www.consultaoviedo.es/	No aplica
A Porta Aberta	Concello da Coruña	https://aportaaberta.coruna.es	https://github.com/ConcelloCoruna/aportaaberta

Problemas de reutilización en el código actual

madrid

Search

We've found 53 code results

Sort: **Best match**

app/mailers/application_mailer.rb — Ruby

Showing the top two matches. Last indexed 12 days ago.

```ruby
class ApplicationMailer < ActionMailer::Base
  helper :settings
  default from: "Decide Madrid <no-reply@madrid.es>"
  layout 'mailer'
end
```

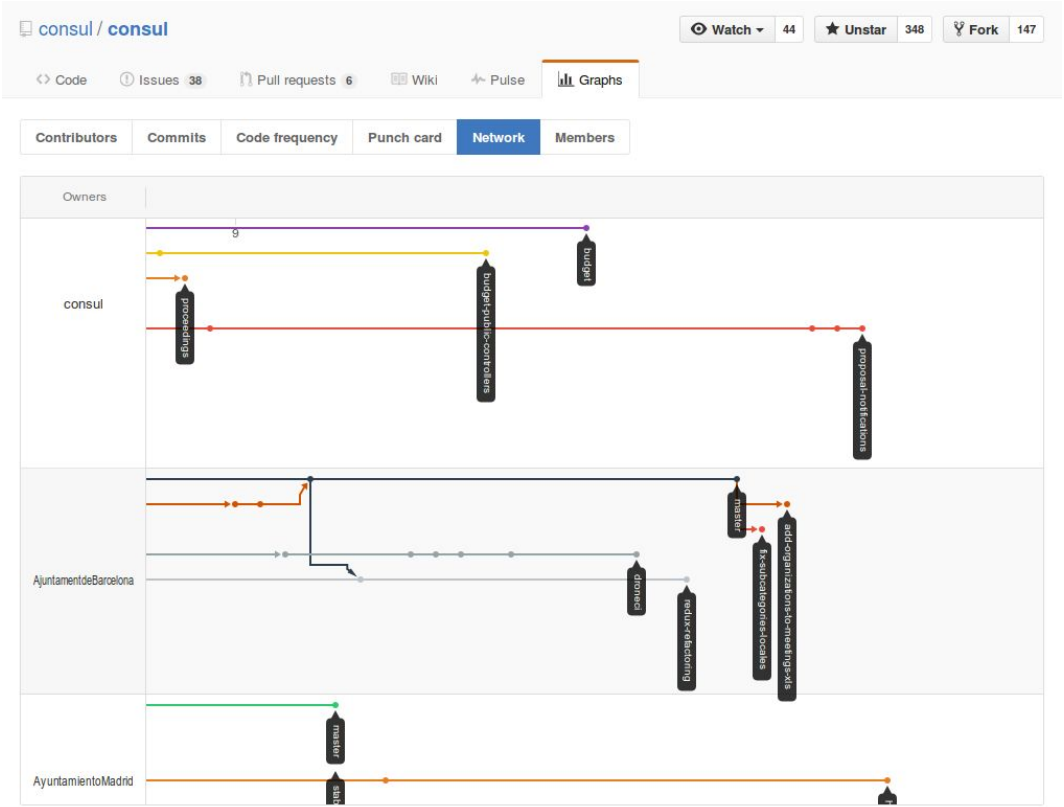
public/sitemap.xml — XML

Showing the top two matches. Last indexed on 28 Mar.

```xml
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>https://decide.madrid.es/</loc>
    <priority>0.7</priority>
  </url>
  <url>
    <loc>https://decide.madrid.es/proposals</loc>
    <priority>0.9</priority>
    <changefreq>always</changefreq>
```

```ruby
99
100   def valid_postal_code?
101     postal_code =~ /^280/
102   end
```

Situación actual

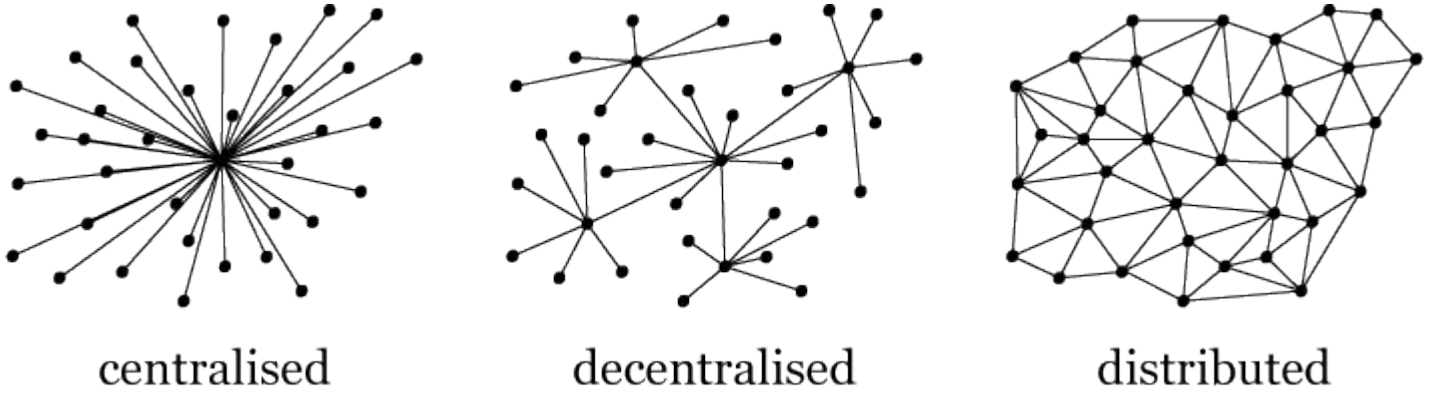


Escenarios

Escenario 1: Desarrollo centralizado de Consul

Escenario 2: Desarrollo descentralizado de Consul

Escenario 3: Desarrollo distribuido de Consul



Escenario 1: Desarrollo centralizado de Consul

Este escenario corresponde al **modelo actual** de funcionamiento: el Ayuntamiento de Madrid mantiene el código de Cónsul, evaluando e incorporando aquellas modificaciones realizadas en los “forks” o bifurcaciones del código que se realicen en otros ayuntamientos u organizaciones.

Escenario 2: Desarrollo descentralizado de Consul

Es posible que las versiones realizadas en unos cuantos ayuntamientos ganen cierta autonomía y generen una cierta **“tipología” de versiones de Consul** cuyo mantenimiento sea realizado por sus impulsores iniciales.

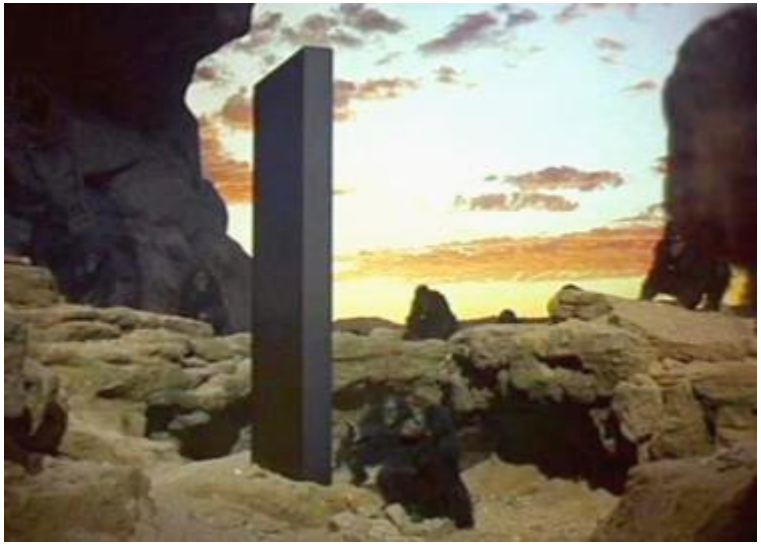
Escenario 3: Desarrollo distribuido de Consul

En el caso de que una **nueva arquitectura** de Consul facilitará la contribución y la reutilización de código, se avanzaría hacia un modelo de desarrollo distribuido.

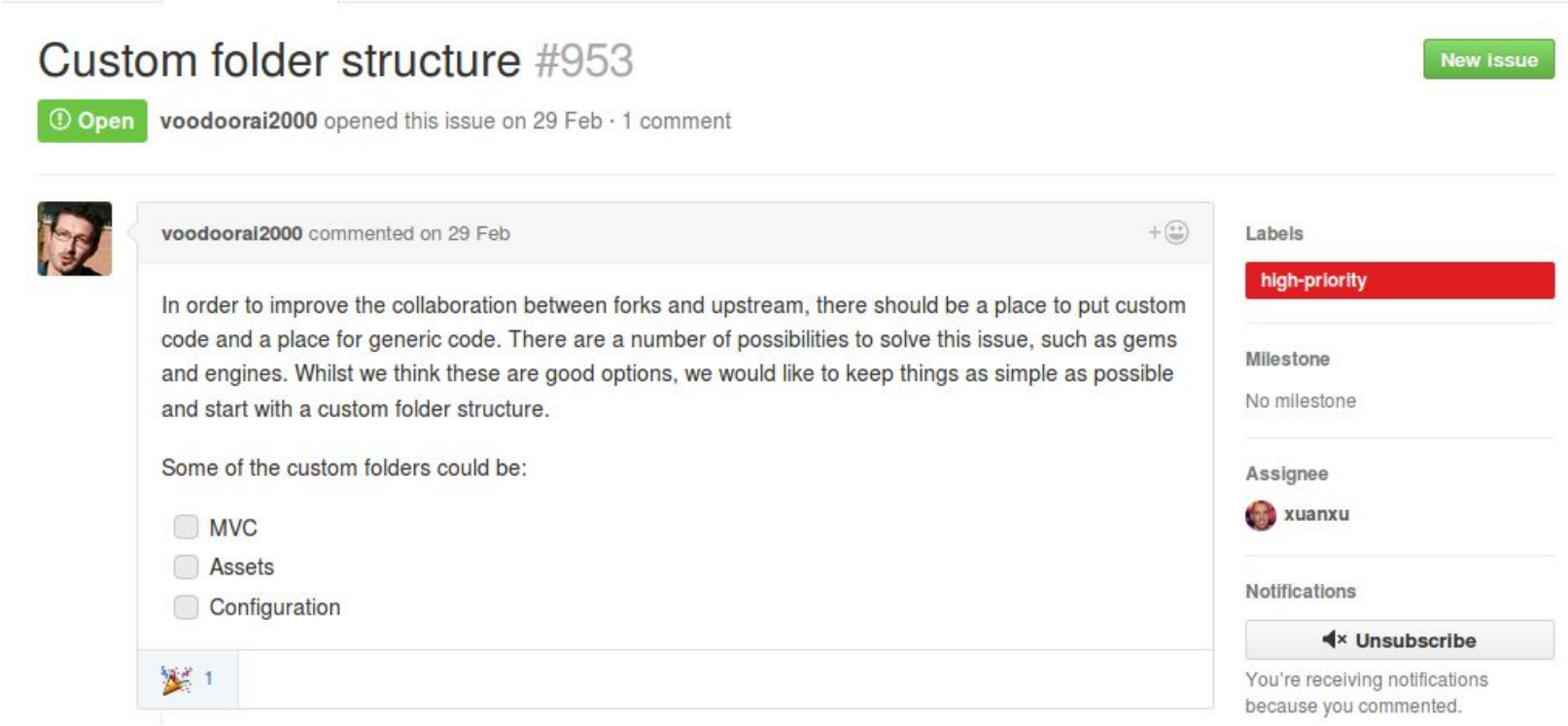
Los **tiempos de desarrollo** al principio se alargarían, aunque a medio plazo posiblemente se reduciría drásticamente la necesidad de escribir código por parte de los municipios que lo quisieran adoptar, y en la mayoría no sería necesario en absoluto.

Alternativa 1: Directorios de personalización

Carpeta donde poner el código específico de cada instalación (diseño, textos, etc).



Alternativa 1: Directorios de personalización



Custom folder structure #953

New issue

Open voodoorai2000 opened this issue on 29 Feb · 1 comment

voodoorai2000 commented on 29 Feb

In order to improve the collaboration between forks and upstream, there should be a place to put custom code and a place for generic code. There are a number of possibilities to solve this issue, such as gems and engines. Whilst we think these are good options, we would like to keep things as simple as possible and start with a custom folder structure.

Some of the custom folders could be:

- [ ] MVC
- [ ] Assets
- [ ] Configuration

1

Labels

high-priority

Milestone

No milestone

Assignee

xuanxu

Notifications

Unsubscribe

You're receiving notifications because you commented.

Alternativa 1: Directorios de personalización

Personalización

Puedes modificar consul y ponerle tu propia imagen, para esto debes primero hacer un fork de https://github.com/consul/consul creando un repositorio nuevo en Github. Puedes usar otro servicio como Gitlab, pero no te olvides de poner el enlace en el footer a tu repositorio en cumplimiento con la licencia de este proyecto (GPL Affero 3).

Hemos creado una estructura específica donde puedes sobreescribir y personalizar la aplicación para que puedas actualizar sin que tengas problemas al hacer merge y se sobreescriban por error tus cambios. Intentamos que Consul sea una aplicación Ruby on Rails lo más plain vanilla posible para facilitar el acceso de nuevas desarrolladoras.

Ficheros y directorios especiales

Para adaptarlo puedes hacerlo a través de los directorios que están en custom dentro de:

- config/locales/custom/
- app/assets/images/custom/
- app/views/custom/
- app/controllers/custom/
- app/models/custom/

https://github.com/consul/consul/blob/master/CUSTOMIZE_ES.md

Alternativa 1: Directorios de personalización

Facilidad de uso

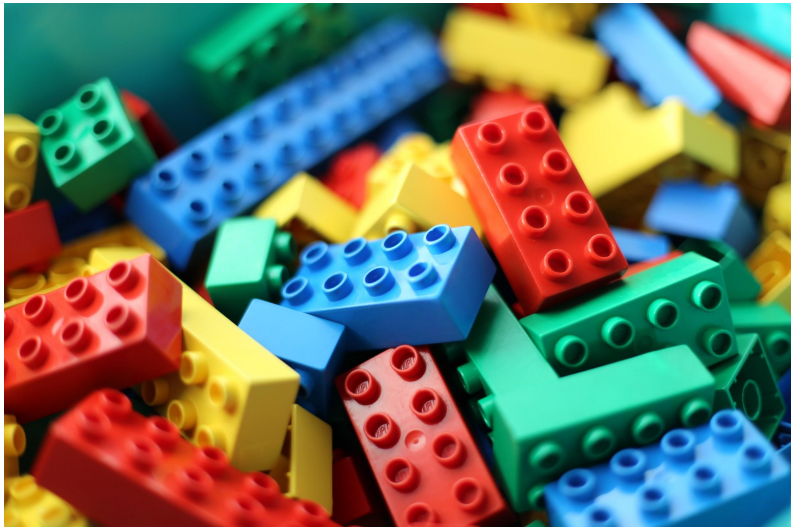
Simplicidad de cara al desarrollo

Impide compartir los distintos módulos de cada proceso diferenciado.

Se **genera un cuello de botella** en la introducción y aceptación de funcionalidades nuevas por parte del equipo de desarrollo de Consul.

Alternativa 2: Modularización (engines)

Mini aplicaciones que proporcionan funcionalidad a sus aplicaciones anfitrión



Alternativa 2: Modularización (engines)

Es más fácil **encontrar un bug**.

Es más fácil **quitar componentes** que no se estén utilizando.

Es más fácil **entender el histórico** del desarrollo de un módulo.

Las **migraciones** se organizan mejor al estar prefijadas con el nombre del engine.

Permite **customizar** mejor las instalaciones.

Es más fácil entender y manejar las **dependencias**.

Provee una forma alternativa de **refactorizar** una funcionalidad siempre y cuando se mantenga la misma API.

Permite un **desarrollo en paralelo** organizado.

Permite **evitar un cuello de botella** en la introducción y aceptación de funcionalidades nuevas.

Alternativa 2: Modularización (engines)

Requiere una **inversión inicial** costosa de tiempo y recursos para el cambio a esta arquitectura, en comparación a otras alternativas.

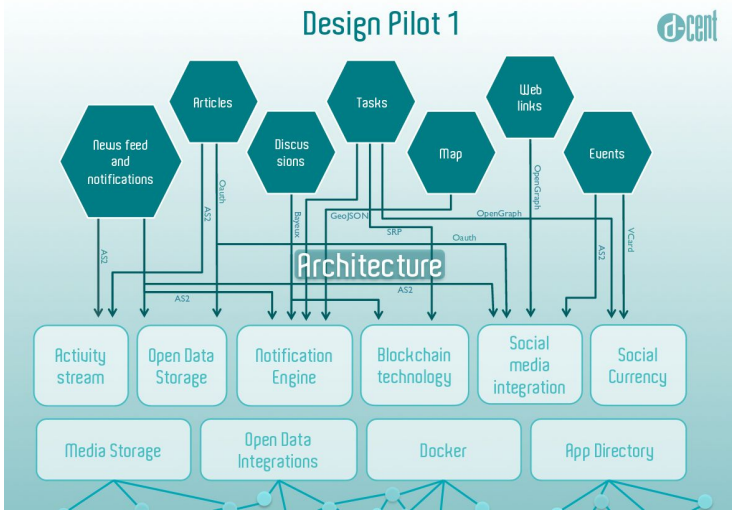
Enlentece la escritura de código. Al empezar con este modelo hay que tener en cuenta en qué componente pertenece cada funcionalidad que se quiera agregar.

Aumenta la **curva de aprendizaje** de gente nueva al proyecto.

Alternativa 3: Microservicios

Enfoque para el desarrollo de una única aplicación como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y comunicándose con mecanismos livianos, a menudo una API HTTP.

http://martinfowler.com/articles/microservices.html



Alternativa 3: Microservicios

Cumple con la mayoría de puntos a favor que se encuentran en la arquitectura “2. Modularización (engines)”.

Su principal ventaja reside en que esta arquitectura permite que cada componente diferenciado esté escrito en un **lenguaje de programación diferente**, por lo que diferentes equipos de desarrollo pueden contribuir sin tener la limitación de que todos tengan que saber Ruby on Rails.

De todas las arquitecturas propuestas es la que **complejiza y enlentece más** el desarrollo.

Aumenta la **curva de aprendizaje** de gente nueva al proyecto.

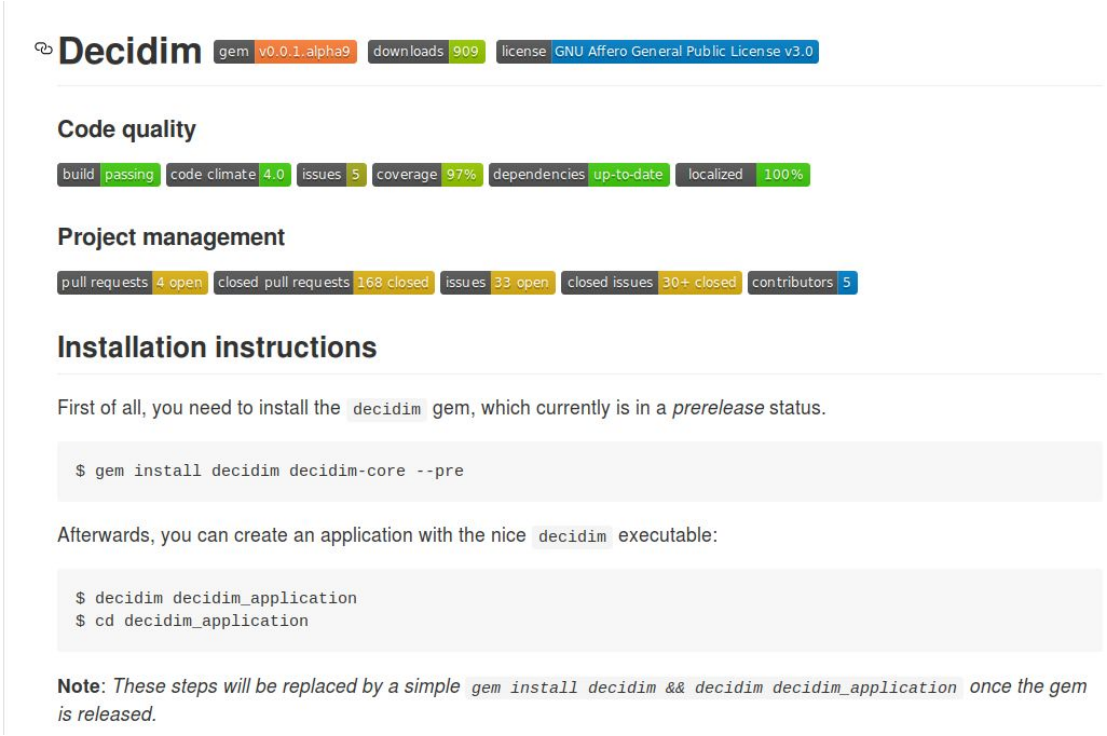
Puede agregar **latencias en la red** por las distintas conexiones que tiene que realizarse para cada petición.

Dificulta tanto el desarrollo como realizar pruebas de integración de todos los servicios y despliegue de los mismos.

Conclusiones

Alternativa	Diseño	Funcional	Traducciones	Compartir	Actualizar	Rapidez
0. Situación actual	NC	NC	NC	NC	NC	C
1. Directorios de personalización	C	P	C	NC	C	C
2. Modularización (engines)	C	C	C	C	C	P
3. Microservicios	C	C	C	C	C	NC

Próximos pasos: nueva versión modular



Decidim

gem v0.0.1.alpha9 | downloads 909 | license GNU Affero General Public License v3.0

Code quality

build passing | code climate 4.0 | issues 5 | coverage 97% | dependencies up-to-date | localized 100%

Project management

pull requests 4 open | closed pull requests 168 closed | issues 33 open | closed issues 30+ closed | contributors 5

Installation instructions

First of all, you need to install the `decidim` gem, which currently is in a *prerelease* status.

```
$ gem install decidim decidim-core --pre
```

Afterwards, you can create an application with the nice `decidim` executable:

```
$ decidim decidim_application
$ cd decidim_application
```

Note: *These steps will be replaced by a simple `gem install decidim && decidim decidim_application` once the gem is released.*