OPEN DATA AT WORK

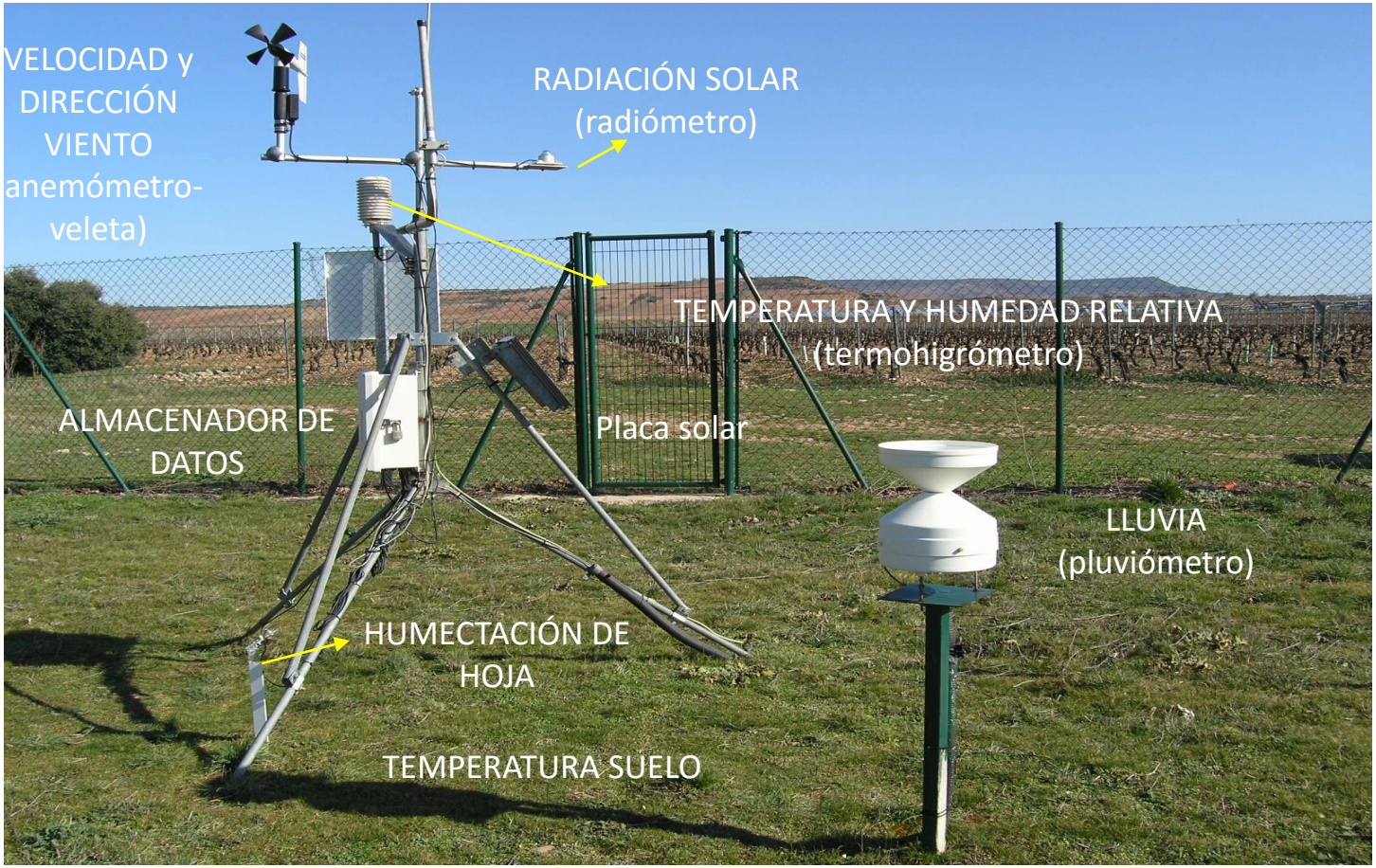
Publicación de datos agroclimáticos a partir de una API REST

Seminario Mirian Andrés, UR, 20 de diciembre de 2016

Vanessa Tobar
Joaquín Huete
Servicio de Información Agroclimática de La Rioja (SIAR)

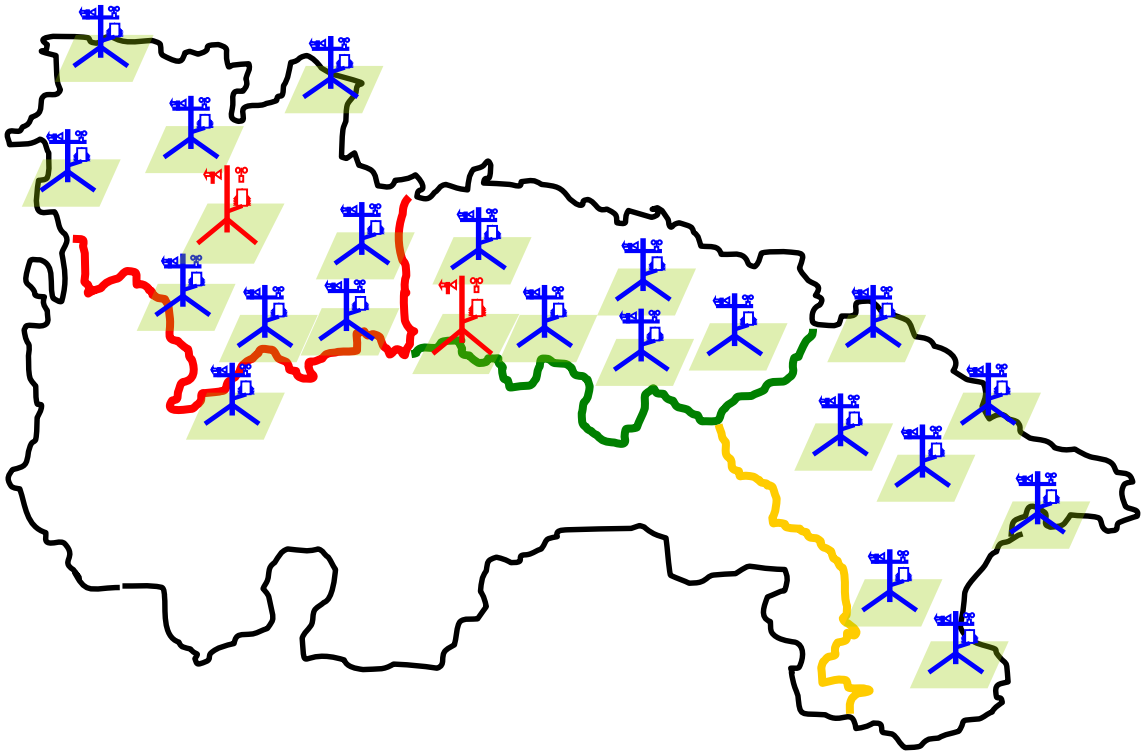
DATOS AGROCLIMÁTICOS

- SIAR: Servicio Información Agroclimática
  - Red estaciones agroclimáticas



DATOS AGROCLIMÁTICOS

21 estaciones en territorio agrícola de La Rioja



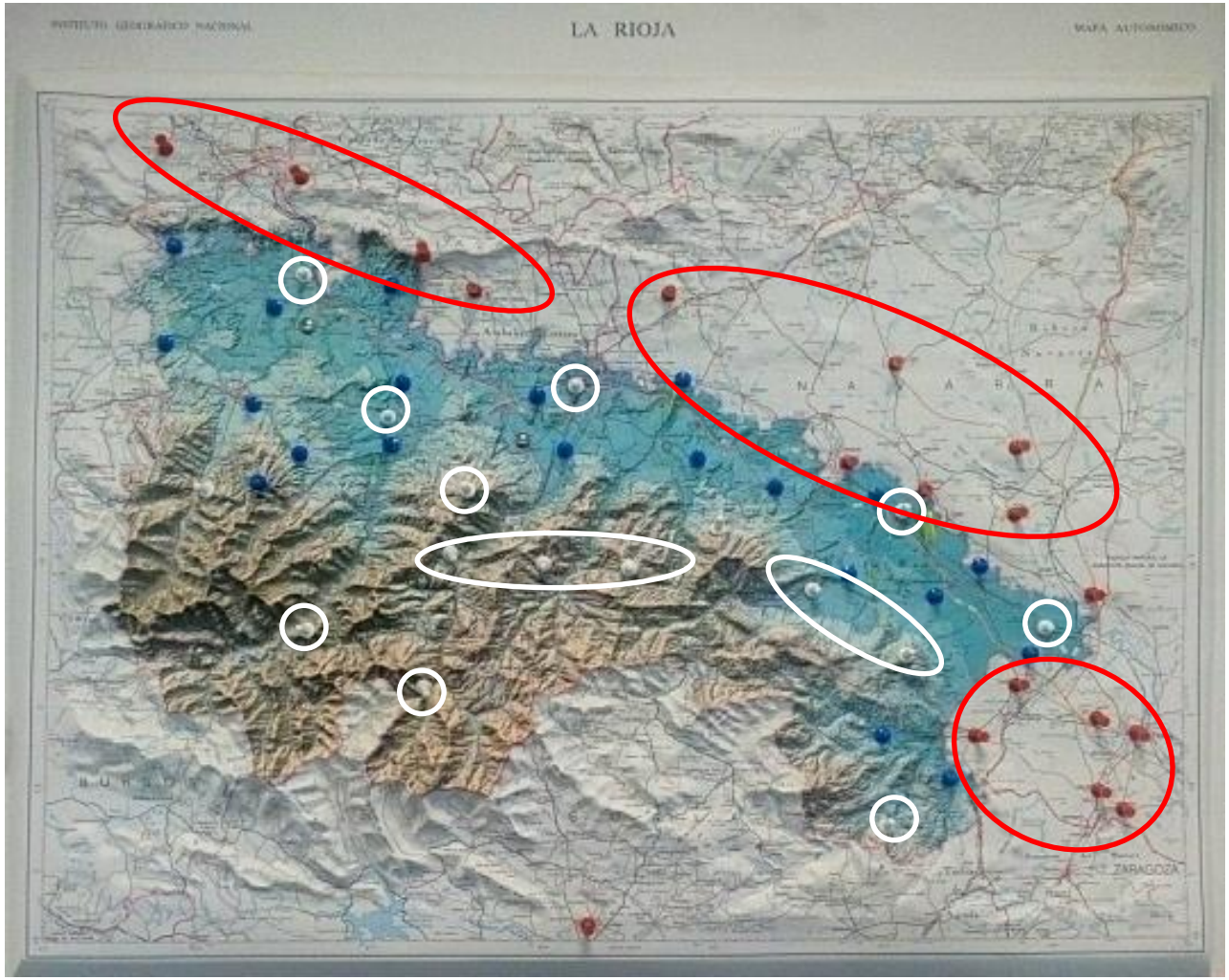
DATOS AGROCLIMÁTICOS

Otras redes en La Rioja y CCAA limítrofes

En azul estaciones SIAR

En blanco estaciones SOS-Rioja

En rojo estaciones CCAA limítrofes



DATOS AGROCLIMÁTICOS

- AMPLIA BASE DE DATOS. Alto interés *científico*
  - Datos cada media hora. 21 parámetros
  - Parámetros climáticos: Tª; HR; Viento; Radiación; Lluvia; Tª suelo; Humectación
  - Datos en tiempo real. Comunicación remota vía GPRS
  - Datos diarios, mensuales
  - Datos semihorarios validados según normativa UNE 500-540

DATOS AGROCLIMÁTICOS VALIDADOS

- Proceso en 2 fases: Automática + Manual
- Analiza la calidad del dato
- Resultado: DATO + METADATO

  Dato agroclimático etiquetado con nivel de validación

DATOS AGROCLIMÁTICOS VALIDADOS

- Fase Automática: 5 test de validación
  - Límites: rígidos y flexibles
  - Coherencia: temporal, interna del dato y de la serie y espacial
- Fase Manual: Inspección Visual

  Confirmación por operador humano del proceso automático

  ⇨ Posibles modificaciones en base de datos “a posteriori”

DATOS AGROCLIMÁTICOS
PRODUCTOS DERIVADOS

- Productos derivados. Alto interés *práctico*
  - Sector Agrícola:
    - Predicción plagas y enfermedades
    - Riego eficiente cultivos
    - Aviso heladas
  - Otros Sectores:
    - Energías Renovables
    - Medio Ambiente: quemas sanitarias de rastrojos
    - Usos civiles: negocios o eventos climáticos en medio agrario
    - Otros

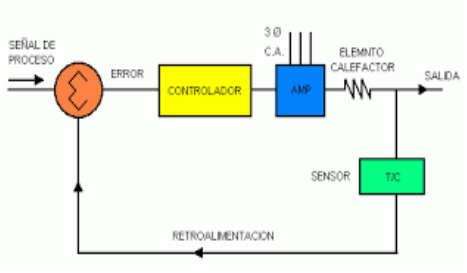
EXPLOTACIÓN DATOS AGROCLIMÁTICOS

- Datos con gran potencial
  - Potencial se multiplica si se integra con otras BBDD. Compartir información
  - Útil para
    - Usuarios externos Administración
    - Usuarios de Administración

⇨ MEJOR SERVICIO AL CIUDADANO

EXPLOTACIÓN DATOS AGROCLIMÁTICOS

- Diversos usos
  - No Agrarios:



  - Agrarios:
    - supuestamente poco tecnificados
    - Agricultura 2.0
    - AgriData (BigData en agricultura)

Permiten contar la historia de una uva que se convierte en un vino admirado en todo el mundo. O te transportan hasta una fría mañana de diciembre entre olivares y ver que **el origen del oro líquido no está en una máquina envasadora, sino en las manos de personas** que aman su trabajo y es su forma de vida.

Los medios sociales permiten jugar bajo las mismas reglas a todos los usuarios, ya sean medios de comunicación, empresas o agricultores y ganaderos subidos a su tractor. **Un espacio en la web donde lo que importa es la originalidad y la creatividad.** Donde una foto con tus vacas puede ser de más interés (viralidad) que la vida de personas encerradas en una casa.

Fuente: Felfies

I Foro Transformación digital y big data en agricultura
29 noviembre 2016
Madrid – Consejo Económico y Social

Inaugura: Ministra MAPAMA "Es necesario hacer uso de todas las tecnología disponibles porque en la innovación está la mejora de nuestra competitividad"

Participan:

Empresas tecnología transformación digital y Big Data: Google, Intel, IBM,…

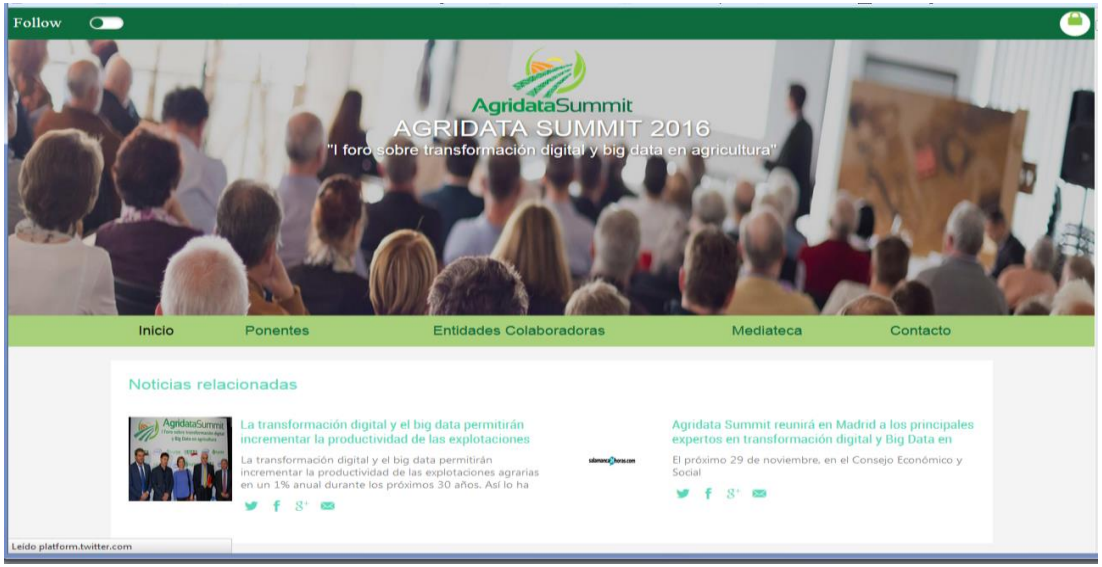
Empresas agrícolas

COAG

Ministerio Agricultura

AEMET

Empresas sistemas de innovación en agricultura



AGRICULTURA 2.0

- Relevo generacional: nuevas necesidades sector agrario. Información instantánea y en tiempo real
- Nuevas tecnologías

- Normativa europea – AGRICULTURA SOSTENIBLE: Necesario datos para justificar decisiones agrícolas
- SMART AGRICULTURE. Agricultura conectada al mundo digital

AGRICULTURA 2.0

- Manejo gran cantidad de información
- Datos procedentes de diferentes BBDD
  - Gran parte está en la Administración

- Herramienta ayuda

Aplicación informática

SENCILLA

apisiar.larioja.org

- OBJETIVO: Ofrecer datos agroclimáticos de forma accesible para consulta y reutilización

- PERMITIRÁ:
  - Publicación datos en cualquier plataforma

    El dispositivo móvil es fundamental en sector agrícola
  - Aportar valor añadido al dato agroclimático

Arquitectura

- Servidor Linux CentOS 6.7 instalado en servidor X86_64 alojado en ARSYS en DMZ externa a www.larioja.org

- Servidor web: apache 2.2.15 integrado con aplicación mediante “Phusion Passenger”

- DNS http://apisiar.larioja.org

Arquitectura

- Base de datos: MariaDB (Server version: 5.5.46-MariaDB)
  - Tablas de datos (dinámicos):
    - Datos climáticos diarios y semihorarios, rango datos
    - Actualizadas cada hora desde servidor en larioja.org
    - Vía Spoon
  - Tablas de configuración (estáticos):
    - Estaciones disponibles, parámetros disponibles, otras.
    - Actualizadas manualmente desde servidor en larioja.org
    - Vía Spoon
  - Tablas de logs:
    - usuarios, tipos de usuarios, accesos a la aplicación
    - Escritas por la aplicación*

Arquitectura

- Aplicación programada en Ruby 1.9.3
  - Principales gemas empleadas
    - Sinatra -> gestiona métodos http (Get, Post, etc)
    - Active record -> gestiona comunicación con BBDD

- Aproximadamente 1800 líneas de código
  - primerserver.rb (900 líneas); gestiona llamadas http
  - parsear.rb (500 líneas de código) gestiona consultas a la bbdd, comprobaciones de campos y parseo a json/csv
  - metodos.rb (300 líneas); gestiona acceso a datos climáticos
  - conexion.rb (100 líneas), configuración de la gema active_record para acceso a las tablas de la BBDD

¿Por qué Ruby?

¿Por qué Ruby?

```ruby
get '/estaciones-disponibles/id/:id' do

    dat1 = Parsear.new

    res = dat1.consultadisponibles()

    disponibles = '{"ESTACIONES":{'
    i=0

    while(i < res.size)
        disponibles = disponibles + '"ESTACION' + (i + 1).to_s +
                                    '":{"CODIGO":"' + res[i]["COD_ESTACION"] +
                                    '","NOMBRE":"' + res[i]["MUNICIPIO"] + '"},'
        i = i + 1
    end

    disponibles = disponibles[0..(disponibles.size-3)]
    disponibles = disponibles + '}}}'

    return disponibles
end
```

¿Por qué Ruby?

```ruby
get '/estaciones-disponibles/id/:id' do

    dat1 = Parsear.new

    res = dat1.consultadisponibles()

    disponibles = '{"ESTACIONES":{'
    i=0

    while(i < res.size)
        disponibles = disponibles + '"ESTACION' + (i + 1).to_s +
                      '":{"CODIGO":"' + res[i]["COD_ESTACION"] +
                      '","NOMBRE":"' + res[i]["MUNICIPIO"] + '"},'
        i = i + 1
    end

    disponibles = disponibles[0..(disponibles.size-3)]
    disponibles = disponibles + '}}}'

    return disponibles
end
```

```ruby
    def consultadisponibles()
        dat=ESTACIONES.select("COD_ESTACION,MUNICIPIO")
        return dat
    end
```

¿Por qué Ruby?

```
535  get '/estaciones-disponibles/id/:id' do
536
537      dat1 = Parsear.new
538
539      res = dat1.consultadisponibles()
540
541      disponibles = '{"ESTACIONES":{'
542
```

```
-bash-4.1$
-bash-4.1$ irb
1.9.3-p551 :001 > require './primerserver.rb'
 => true
1.9.3-p551 :002 > dat1 = Parsear.new
 => #<Parsear:0x00000003a04e68>
1.9.3-p551 :003 > res = dat1.consultadisponibles()
D, [2016-12-19T10:54:26.674321 #11166] DEBUG -- :   ESTACIONES Load (0.9ms)  SEL
ECT COD_ESTACION,MUNICIPIO FROM `ESTACIONES`
 => #<ActiveRecord::Relation [#<ESTACIONES COD_ESTACION: "501", MUNICIPIO: "AGON
CILLO">, #<ESTACIONES COD_ESTACION: "502", MUNICIPIO: "Aldeanueva de Ebro">, #<E
STACIONES COD_ESTACION: "503", MUNICIPIO: "Santo Domingo">, #<ESTACIONES COD_EST
ACION: "504", MUNICIPIO: "VILLAR DE TORRE">, #<ESTACIONES COD_ESTACION: "505", M
UNICIPIO: "CASALARREINA">, #<ESTACIONES COD_ESTACION: "506", MUNICIPIO: "ALFARO"
>, #<ESTACIONES COD_ESTACION: "507", MUNICIPIO: "TORREMONTALBO">, #<ESTACIONES C
OD_ESTACION: "508", MUNICIPIO: "RINCON DE SOTO">, #<ESTACIONES COD_ESTACION: "50
9", MUNICIPIO: "Logro\xF1o">, #<ESTACIONES COD_ESTACION: "510", MUNICIPIO: "San
Vicente">, ...]>
1.9.3-p551 :004 > res.size
 => 22
1.9.3-p551 :005 >
```

¿Por qué Ruby?

```
535  get '/estaciones-disponibles/id/:id' do
536
537      dat1 = Parsear.new
538
539      res = dat1.consultadisponibles()
540
541      disponibles = '{"ESTACIONES":{'
542
543  -bash-4.1$
544  -bash-4.1$ irb
545  1.9.3-p551 :001 > require './primerserver.rb'
546   => true
547  1.9.3-p551 :002 > dat1 = Parsear.new
548   => #<Parsear:0x00000003a04e68>
549  1.9.3-p551 :003 > res = dat1.consultadisponibles()
550  D, [2016-12-19T10:54:26.674321 #11166] DEBUG -- :   ESTACIONES Load (0.9ms)  SEL
551  ECT COD_ESTACION,MUNICIPIO FROM `ESTACIONES`
552   => #<ActiveRecord::Relation [#<ESTACIONES COD_ESTACION: "501", MUNICIPIO: "AGON
553  CILLO">, #<ESTACIONES COD_ESTACION: "502", MUNICIPIO: "Aldeanueva de Ebro">, #<E
554  STACIONES COD_ESTACION: "503", MUNICIPIO: "Santo Domingo">, #<ESTACIONES COD_EST
555  ACION: "504", MUNICIPIO: "VILLAR DE TORRE">, #<ESTACIONES COD_ESTACION: "505", M
     UNICIPIO: "CASALARREINA">, #<ESTACIONES COD_ESTACION: "506", MUNICIPIO: "ALFARO"
     >, #<ESTACIONES COD_ESTACION: "507", MUNICIPIO: "TORREMONTALBO">, #<ESTACIONES C
     OD_ESTACION: "508", MUNICIPIO: "RINCON DE SOTO">, #<ESTACIONES COD_ESTACION: "50
     9", MUNICIPIO: "Logro\xF1o">, #<ESTACIONES COD_ESTACION: "510", MUNICIPIO: "San
     Vicente">, ...]>
     1.9.3-p551 :004 > res.size
      => 22
     1.9.3-p551 :005 >
```

¿Por qué Ruby?

```
535  get '/estaciones-disponibles/id/:id' do
536
537      dat1 = Parsear.new
538
539      res = dat1.consultadisponibles()
540
541      disponibles = '{"ESTACIONES":{'
542
```

```
-bash-4.1$
-bash-4.1$ irb
1.9.3-p551 :001 > require './primerserver.rb'
 => true
1.9.3-p551 :002 > dat1 = Parsear.new
 => #<Parsear:0x00000003a04e68>
1.9.3-p551 :003 > res = dat1.consultadisponibles()
D, [2016-12-19T10:54:26.674321 #11166] DEBUG -- :   ESTACIONES Load (0.9ms)  SEL
ECT COD_ESTACION,MUNICIPIO FROM `siarapi`.`ESTACIONES`
 => #<ActiveRecord::Relation [#<ESTACIONES COD_ESTACION: "501", MUNICIPIO: "AGON
CILLO">, #<ESTACIONES COD_ESTACION: "502", MUNICIPIO: "Aldeanueva de Ebro">, #<E
STACIONES COD_ESTACION: "503", MUNICIPIO: "Santo Domingo">, #<ESTACIONES COD_EST
ACION: "504", MUNICIPIO: "VILLAR DE TORRE">, #<ESTACIONES COD_ESTACION: "505", M
UNICIPIO: "CASALARREINA">, #<ESTACIONES COD_ESTACION: "506", MUNICIPIO: "ALFARO"
>, #<ESTACIONES COD_ESTACION: "507", MUNICIPIO: "TORREMONTALBO">, #<ESTACIONES C
OD_ESTACION: "508", MUNICIPIO: "RINCON DE SOTO">, #<ESTACIONES COD_ESTACION: "50
9", MUNICIPIO: "Logro\xF1o">, #<ESTACIONES COD_ESTACION: "510", MUNICIPIO: "San
Vicente">, ...]>
1.9.3-p551 :004 > res.size
 => 22
1.9.3-p551 :005 >
```

¿Por qué Ruby?

```
535  get '/estaciones-disponibles/id/:id' do
536
537      dat1 = Parsear.new
538
539      res = dat1.consultadisponibles()
540
541      disponibles = '{"ESTACIONES":{'
542
543  -bash-4.1$
544  -bash-4.1$ irb
545  1.9.3-p551 :001 > require './primerserver.rb'
546   => true
547  1.9.3-p551 :002 > dat1 = Parsear.new
548   => #<Parsear:0x00000003a04ef8>
549  1.9.3-p551 :003 > res = dat1.consultadisponibles()
550  D, [2016-12-19T10:54:26.674321 #11166] DEBUG -- :   ESTACIONES Load (0.9ms)  SEL
551  ECT COD_ESTACION,MUNICIPIO FROM `siarapi`.`ESTACIONES`
552   => #<ActiveRecord::Relation [#<ESTACIONES COD_ESTACION: "501", MUNICIPIO: "AGON
553  CILLO">, #<ESTACIONES COD_ESTACION: "502", MUNICIPIO: "Aldeanueva de Ebro">, #<E
554  STACIONES COD_ESTACION: "503", MUNICIPIO: "Santo Domingo">, #<ESTACIONES COD_EST
555  ACION: "504", MUNICIPIO: "VILLAR DE TORRE">, #<ESTACIONES COD_ESTACION: "505", M
     UNICIPIO: "CASALARREINA">, #<ESTACIONES COD_ESTACION: "506", MUNICIPIO: "ALFARO"
     >, #<ESTACIONES COD_ESTACION: "507", MUNICIPIO: "TORREMONTALBO">, #<ESTACIONES C
     OD_ESTACION: "508", MUNICIPIO: "RINCON DE SOTO">, #<ESTACIONES COD_ESTACION: "50
     9", MUNICIPIO: "Logro\xF1o">, #<ESTACIONES COD_ESTACION: "510", MUNICIPIO: "San
     Vicente">, ...]>
     1.9.3-p551 :004 > res.size
      => 22
     1.9.3-p551 :005 >
```

¿Por qué Ruby?

```
535  get '/estaciones-disponibles/id/:id' do
536
537      dat1 = Parsear.new
538
539      res = dat1.consultadisponibles()
540
541      disponibles = '{"ESTACIONES":{'
542
```

```
-bash-4.1$
-bash-4.1$ irb
1.9.3-p551 :001 > require './primerserver.rb'
 => true
1.9.3-p551 :002 > dat1 = Parsear.new
 => #<Parsear:0x00000003a04e68>
1.9.3-p551 :003 > res = dat1.consultadisponibles()
D, [2016-12-19T10:54:26.674321 #11166] DEBUG -- :   ESTACIONES Load (0.9ms)  SEL
ECT COD_ESTACION,MUNICIPIO FROM [redacted]ESTACIONES`
 => #<ActiveRecord::Relation [#<ESTACIONES COD_ESTACION: "501", MUNICIPIO: "AGON
CILLO">, #<ESTACIONES COD_ESTACION: "502", MUNICIPIO: "Aldeanueva de Ebro">, #<E
STACIONES COD_ESTACION: "503", MUNICIPIO: "Santo Domingo">, #<ESTACIONES COD_EST
ACION: "504", MUNICIPIO: "VILLAR DE TORRE">, #<ESTACIONES COD_ESTACION: "505", M
UNICIPIO: "CASALARREINA">, #<ESTACIONES COD_ESTACION: "506", MUNICIPIO: "ALFARO"
>, #<ESTACIONES COD_ESTACION: "507", MUNICIPIO: "TORREMONTALBO">, #<ESTACIONES C
OD_ESTACION: "508", MUNICIPIO: "RINCON DE SOTO">, #<ESTACIONES COD_ESTACION: "50
9", MUNICIPIO: "Logro\xF1o">, #<ESTACIONES COD_ESTACION: "510", MUNICIPIO: "San
Vicente">, ...]>
1.9.3-p551 :004 > res.size
 => 22
1.9.3-p551 :005 >
```

¿Por qué Ruby?

```
535  get '/estaciones-disponibles/id/:id' do
536
537      dat1 = Parsear.new
538
539      res = dat1.consultadisponibles()
540
541      disponibles = '{"ESTACIONES":{'
542
```

```
-bash-4.1$
-bash-4.1$ irb
1.9.3-p551 :001 > require './primerserver.rb'
 => true
1.9.3-p551 :002 > dat1 = Parsear.new
 => #<Parsear:0x00000003a04e68>
1.9.3-p551 :003 > res = dat1.consultadisponibles()
D, [2016-12-19T10:54:26.674321 #[illegible]] DEBUG -- :   ESTACIONES Load (0.9ms)  SEL
ECT COD_ESTACION,MUNICIPIO FROM [redacted]ESTACIONES`
 => #<ActiveRecord::Relation [#<ESTACIONES COD_ESTACION: "501", MUNICIPIO: "AGON
CILLO">, #<ESTACIONES COD_ESTACION: "502", MUNICIPIO: "Aldeanueva de Ebro">, #<E
STACIONES COD_ESTACION: "503", MUNICIPIO: "Santo Domingo">, #<ESTACIONES COD_EST
ACION: "504", MUNICIPIO: "VILLAR DE TORRE">, #<ESTACIONES COD_ESTACION: "505", M
UNICIPIO: "CASALARREINA">, #<ESTACIONES COD_ESTACION: "506", MUNICIPIO: "ALFARO"
>, #<ESTACIONES COD_ESTACION: "507", MUNICIPIO: "TORREMONTALBO">, #<ESTACIONES C
OD_ESTACION: "508", MUNICIPIO: "RINCON DE SOTO">, #<ESTACIONES COD_ESTACION: "50
9", MUNICIPIO: "Logro\xF1o">, #<ESTACIONES COD_ESTACION: "510", MUNICIPIO: "San
Vicente">, ...]>
1.9.3-p551 :004 > res.size
 => 22
1.9.3-p551 :005 >
```

API

- Métodos de acceso a datos estáticos (preconfiguración)
- Métodos de acceso a datos dinámicos
- Seguridad de la aplicación
- Escritura de logs

API

- Métodos de acceso a datos estáticos (preconfiguración)

http://apisiar.larioja.org +

/ultima-hora/TU_ID

/estaciones-disponibles/id/TU_ID

/estacion/COD_ESTACION/parametros-disponibles/id/TU_ID

/rango-datos/estacion/COD_ESTACION/tipodato/TIPO_DATO/id/TU_ID

API

- Métodos de acceso a datos dinámicos (extracción de datos)

http://apisiar.larioja.org +

/estacion/COD_ESTACION/tipo/TIPO_DATO/fechain/AAAAMMDD
/fechafin/AAAAMMDD/salida/TIPO_SALIDA/niveles/SI-NO
/id/TU_ID

/estacion/COD_ESTACION/tipo/semihorario/fechain/AAAAMMDD
/horain/HHMM/fechafin/AAAAMMDD/horafin/HHMM
/salida/TIPO_SALIDA/niveles/SI-NO/id/TU_ID

API

- Seguridad de la aplicación
  - Hay un máximo número de conexiones activas que el servidor gestiona
  - Todas las consultas exigen una ID activa
  - Las consultas que no se ajustan a los patrones son redirigidas al index
  - Los errores en el servidor son enmascarados al usuario

```ruby
before '/*/id/:id' do

    # Variables de control
    $numeroconex = $numeroconex + 1 # máximo número de conexiones activas
    @conexionactiva = $numeroconex
    @t1 = Time.now
    dat = Parsear.new
    userid = params[:id].to_s

    # Comprobacion del número máximo de conexiones activas
    if($numeroconex >= NUM_MAX_CONEXIONES_ACTIVAS)
        $numeroconex = $numeroconex - 1
        redirect '/servidor-saturado'
    end

    res = dat.consultaactivo(userid)
    if (res[0].inspect == "nil")
        @id = 9999 # Si se deniega se usa esta ID para apuntar el error en API_LOGS
        redirect '/denegado'
    elsif(res[0]["ACTIVO"] == "n")
        @id = 9998 # Si el usuario está inactivado se usa esta ID para apuntar el error en API_LOGS
        redirect '/usuarioInactivo'
    end

    @id = res[0]["id"]
    msg=""
end
```

```ruby
before '/*/id/:id' do

    # Variables de control
    $numeroconex = $numeroconex + 1 # máximo número de conexiones activas
    @conexionactiva = $numeroconex
    @t1 = Time.now
    dat = Parsear.new
    userid = params[:id].to_s

    # Comprobacion del número máximo de conexiones activas
    if($numeroconex >= NUM_MAX_CONEXIONES_ACTIVAS)
        $numeroconex = $numeroconex - 1
        redirect '/servidor-saturado'
    end

    res = dat.consultaactivo(userid)
    if (res[0].inspect == "nil")
        @id = 9999 # Si se deniega se usa esta ID para apuntar el error en API_LOGS
        redirect '/denegado'
    elsif(res[0]["ACTIVO"] == "n")
        @id = 9998 # Si el usuario está inactivado se usa esta ID para apuntar el error en API_LOGS
        redirect '/usuarioInactivo'
    end

    @id = res[0]["id"]
    msg=""
end
```

API

- Seguridad de la aplicación
  - Hay un máximo número de conexiones activas que el servidor gestiona
  - Todas las consultas exigen una ID válida y activa
  - Las consultas que no se ajustan a los patrones son redirigidas al index
  - Los errores en el servidor son enmascarados al usuario

```ruby
before '/*/id/:id' do

    # Variables de control
    $numeroconex = $numeroconex + 1 # máximo número de conexiones activas
    @conexionactiva = $numeroconex
    @t1 = Time.now
    dat = Parsear.new
    userid = params[:id].to_s

    # Comprobacion del número máximo de conexiones activas
    if($numeroconex >= NUM_MAX_CONEXIONES_ACTIVAS)
        $numeroconex = $numeroconex - 1
        redirect '/servidor-saturado'
    end

    res = dat.consultaactivo(userid)
    if (res[0].inspect == "nil")
        @id = 9999 # Si se deniega se usa esta ID para apuntar el error en API_LOGS
        redirect '/denegado'
    elsif(res[0]["ACTIVO"] == "n")
        @id = 9998 # Si el usuario está inactivado se usa esta ID para apuntar el error en API_LOGS
        redirect '/usuarioInactivo'
    end

    @id = res[0]["id"]
    msg=""
end
```

```ruby
get '/servidor-saturado' do
    #ActiveRecord::Base.clear_active_connections!
    ActiveRecord::Base.connection.close

    dat1=Parsear.new
    errores = Array.new

        msg = "Lo sentimos el servicio esta saturado, intentelo de nuevo mas tarde."
        errores.push(msg)

        error = dat1.gestionaerror(4,errores)

        File.open("Error.txt", "w") do |f1|
            f1.puts(error)
        end
    send_file("Error.txt")
end
```

API

- Seguridad de la aplicación
  - Hay un máximo número de conexiones activas que el servidor gestiona
  - Todas las consultas exigen una ID válida y activa

    http://apisiar.larioja.org/ultima-hora/id/INVENTADA
  - Las consultas que no se ajustan a los patrones son redirigidas al index
  - Los errores en el servidor son enmascarados al usuario

```ruby
before '/*/id/:id' do

    # Variables de control
    $numeroconex = $numeroconex + 1 # máximo número de conexiones activas
    @conexionactiva = $numeroconex
    @t1 = Time.now
    dat = Parsear.new
    userid = params[:id].to_s

    # Comprobacion del número máximo de conexiones activas
    if($numeroconex >= NUM_MAX_CONEXIONES_ACTIVAS)
        $numeroconex = $numeroconex - 1
        redirect '/servidor-saturado'
    end

    res = dat.consultaactivo(userid)
    if (res[0].inspect == "nil")
        @id = 9999 # Si se deniega se usa esta ID para apuntar el error en API_LOGS
        redirect '/denegado'
    elsif(res[0]["ACTIVO"] == "n")
        @id = 9998 # Si el usuario está inactivado se usa esta ID para apuntar el error en API_LOGS
        redirect '/usuarioInactivo'
    end

    @id = res[0]["id"]
    msg=""
end
```

```ruby
before '/*/id/:id' do

    # Variables de control
    $numeroconex = $numeroconex + 1 # máximo número de conexiones activas
    @conexionactiva = $numeroconex
    @t1 = Time.now
    dat = Parsear.new
    userid = params[:id].to_s

    # Comprobacion del número máximo de conexiones activas
    if($numeroconex >= NUM_MAX_CONEXIONES_ACTIVAS)
        $numeroconex = $numeroconex - 1
        redirect '/servidor-saturado'
    end

    res = dat.consultaactivo(userid)
    if (res[0].inspect == "nil")
        @id = 9999 # Si se deniega se usa esta ID para apuntar el error en API_LOGS
        redirect '/denegado'
    elsif(res[0]["ACTIVO"] == "n")
        @id = 9998 # Si el usuario está inactivo se usa esta ID para apuntar el error en API_LOGS
        redirect '/usuarioInactivo'
    end

    @id = res[0]["id"]
    msg=""
end
```

```ruby
    def consultaactivo(id)
        res = USUARIOS.where("USER_ID=?",id)
        return res
    end
```

```ruby
get '/denegado' do
    # Gestiona el caso de que un usuario intente un acceso a la API sin una ID válida
    dat1=Parsear.new
    errores = Array.new

        msg = "El usuario no esta dado de alta, por favor contacte con el SIAR (siar.
        errores.push(msg)

        error = dat1.gestionaerror(3,errores)

        File.open("Error.txt", "w") do |f1|
            f1.puts(error)
        end

    send_file("Error.txt")
end
```

API

- Seguridad de la aplicación
  - Hay un máximo número de conexiones activas que el servidor gestiona
  - Todas las consultas exigen una ID válida y activa

    http://apisiar.larioja.org/ultima-hora/id/INVENTADA

    http://apisiar.larioja.org/ultima-hora/id/ID_NO_ACTIVA
  - Las consultas que no se ajustan a los patrones son redirigidas al index
  - Los errores en el servidor son enmascarados al usuario

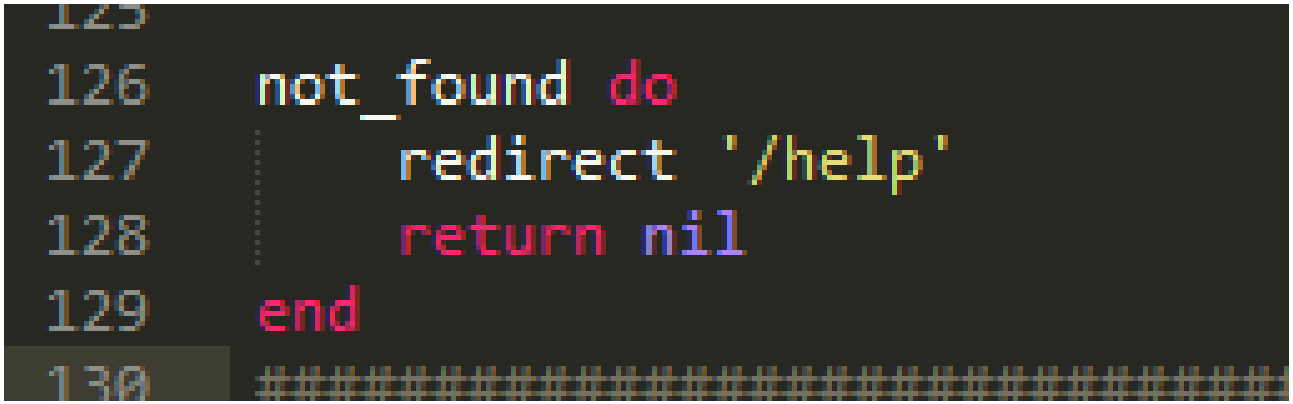
```ruby
get '/usuarioInactivo' do
    # Gestiona el caso de que un usuario intente un acceso a la API sin una ID válida
    dat1=Parsear.new
    errores = Array.new

        msg = "Este usuario esta inactivado, por favor contacte con el SIAR (siar.cida@la
        errores.push(msg)

        error = dat1.gestionaerror(3,errores)

        File.open("Error.txt", "w") do |f1|
            f1.puts(error)
        end

    send_file("Error.txt")
end
```

API

- Seguridad de la aplicación
  - Hay un máximo número de conexiones activas que el servidor gestiona
  - Todas las consultas exigen una ID válida y activa

    http://apisiar.larioja.org/ultima-hora/id/INVENTADA

    http://apisiar.larioja.org/ultima-hora/id/ID_NO_ACTIVA
  - Las consultas que no se ajustan a los patrones son redirigidas al index
  - Los errores en el servidor son enmascarados al usuario


125
126 not_found do
127 redirect '/help'
128 return nil
129 end
130 ##

API

- Seguridad de la aplicación
  - Hay un máximo número de conexiones activas que el servidor gestiona
  - Todas las consultas exigen una ID válida y activa

    http://apisiar.larioja.org/ultima-hora/id/INVENTADA

    http://apisiar.larioja.org/ultima-hora/id/ID_NO_ACTIVA
  - Las consultas que no se ajustan a los patrones son redirigidas al index
  - Los errores en el servidor son enmascarados al usuario

API

```
131  error 500 do
132      error = '{"ERRORES":{"CODIGO":"4","Mensaje0":"Parece que tenemos algunos problemas"}}'
133
134      File.open("Error.txt", "w") do |f1|
135          f1.puts(error)
136      end
137      send_file("Error.txt")
138  end
```

API

- Escritura de logs
  - Al devolverse el resultado de la consulta al usuario la aplicación pasa por un filtro (método AFTER)

```ruby
after '/*/id/:id' do

    dat=Parsear.new
    t2 = Time.now
    url = request.fullpath
    res = response.status
    log = ""
    @tiempo = t2 - @t1

    log = LOGS.new
    puts(LOGS.column_names)
    #log.id=3
    log.USUARIO_ID = @id
    log.CONSULTA = url
    log.RESULTADO = res
    log.TIEMPO_RESPUESTA = @tiempo
    log.FECHA = t2

    i = 0
    while i < 5
        begin
            log.save
            break
        rescue
            sleep(2)
            i = i + 1
        end
    end

    if(i == 5)
        redirect '/servidor-saturado2'
    end

    if(res.to_s == "500")

        error = '{"ERRORES":{"CODIGO":"4","Mensaje0":"Parece que tenemos algunos problemas"}}'
        File.open("Error.txt", "w") do |f1|
            f1.puts(error)
        end
        send_file("Error.txt")
    end

    $numeroconex = $numeroconex - 1
    ActiveRecord::Base.connection.close
end
```

API

- Escritura de logs
  - Al devolverse el resultado de la consulta al usuario la aplicación pasa por un filtro (método AFTER)
  - Almacenamos quién ha accedido, qué ha solicitado, el tiempo que se ha tardado en responder y si la respuesta ha sido satisfactoria.

```ruby
after '/*/id/:id' do

    dat=Parsear.new
    t2 = Time.now
    url = request.fullpath
    res = response.status
    log = ""
    @tiempo = t2 - @t1

    log = LOGS.new
    puts(LOGS.column_names)
    #log.id=3
    log.USUARIO_ID = @id
    log.CONSULTA = url
    log.RESULTADO = res
    log.TIEMPO_RESPUESTA = @tiempo
    log.FECHA = t2

    i = 0
    while i < 5
        begin
            log.save
            break
        rescue
            sleep(2)
            i = i + 1
        end
    end

    if(i == 5)
        redirect '/servidor-saturado2'
    end

    if(res.to_s == "500")

        error = '{"ERRORES":{"CODIGO":"4","Mensaje0":"Parece que tenemos algunos problemas"}}'
        File.open("Error.txt", "w") do |f1|
            f1.puts(error)
        end
        send_file("Error.txt")
    end

    $numeroconex = $numeroconex - 1
    ActiveRecord::Base.connection.close
end
```

API

- Ejemplo de método para acceso a datos

```ruby
# METODO GENERAL PARA EXTRAER DATOS COMPLETOS, COMPROBAR EL HORARIO, CREO QUE HABRÍA QUE PONER LA SEGUNDA HORA EN
get '/estacion/:numero/tipo/:tipodat/fechain/:fecha1/fechafin/:fecha2/salida/:salida/niveles/:niveles/id/:id' do


    fallos= Array.new
    dat1 = Parsear.new #Creamos el objeto para poder llamar a los métodos de consulta y parseo.

    # COMPROBAR QUE LOS PARAMETROS INTRODUCIDOS SON LOS ESPERABLES
    fallos= dat1.compruebacampos(params[:numero].to_i,params[:tipodat].to_s,
                                 params[:fecha1],
                                 params[:fecha2],
                                 "sinhora",
                                 "sinhora",
                                 params[:salida].to_s)

    dat = Metodos.new

    # SI HAY ERRORES ENTONCES SE GESTIONA EL ERROR, SINO SE REALIZA LA PETICION
    if(fallos.size > 0)
        error=dat1.gestionaerror(1,fallos)

        File.open("Error.txt", "w") do |f1|
            f1.puts(error)
        end

        send_file("Error.txt")
    else
        res = dat.metodogeneral4(params[:numero].to_i,      params[:tipodat].to_s,
                                 params[:fecha1],           "0000",
                                 params[:fecha2],           "2330",
                                 "todos",                   params[:salida].to_s,
                                 params[:niveles].to_s)
    end

    File.open("Datos.txt", "w") do |f1|
        f1.puts(res)
    end

    send_file("Datos.txt")
end
################################################################################
```

API

- Ejemplo de método para acceso a datos
  - Define la estructura que tendrá la petición http en la primera línea del código
  - Se realiza un filtrado de campos para analizar si la solicitud es válida
  - Si todo es OK se realiza la petición a la BBDD
  - Por último se genera un archivo con el resultado que es lo que finalmente se pasa al usuario

```
264  # METODO GENERAL PARA EXTRAER DATOS COMPLETOS, COMPROBAR EL HORARIO, CREO QUE HABRIA QUE PONER LA SEGUNDA HORA EN
265  get '/estacion/:numero/tipo/:tipodat/fechain/:fecha1/fechafin/:fecha2/salida/:salida/niveles/:niveles/id/:id' do
266
267
268      fallos= Array.new
269      dat1 = Parsear.new #Creamos el objeto para poder llamar a los métodos de consulta y parseo.
270
271      # COMPROBAR QUE LOS PARAMETROS INTRODUCIDOS SON LOS ESPERABLES
272      fallos= dat1.compruebacampos(params[:numero].to_i,params[:tipodat].to_s,
273                                   params[:fecha1],
274                                   params[:fecha2],
275                                   "sinhora",
276                                   "sinhora",
277                                   params[:salida].to_s)
278
279      dat = Metodos.new
280
281      # SI HAY ERRORES ENTONCES SE GESTIONA EL ERROR, SINO SE REALIZA LA PETICION
282      if(fallos.size > 0)
283          error= dat1.gestionaerror(1,fallos)
284
285          File.open("Error.txt", "w") do |f1|
286              f1.puts(error)
287          end
288
289          send_file("Error.txt")
290      else
291          res = dat.metodogeneral4(params[:numero].to_i,      params[:tipodat].to_s,
292                                   params[:fecha1],           "0000",
293                                   params[:fecha2],           "2330",
294                                   "todos",                   params[:salida].to_s,
295                                   params[:niveles].to_s)
296      end
297
298      File.open("Datos.txt", "w") do |f1|
299          f1.puts(res)
300      end
301
302      send_file("Datos.txt")
303  end
304  ################################################################################
```

RETOS INFORMÁTICOS

- Pequeñas mejoras en la aplicación (hay bastantes detalles):
  - Mejorar las estructuras JSON enviadas, casi todas válidas y bien formadas pero todas mejorables

http://apisiar.larioja.org/ultima-hora/id/TU_ID


```
{
        "DATOS": {
                "DATOS0": 21: {
                        "Estacion": "501",
                        "Fecha-Hora": "2016-12-01T12:30:00.000Z"
                },
                "DATOS1": 21: {
                        "Estacion": "502",
                        "Fecha-Hora": "2016-12-01T12:30:00.000Z"
                },
...
```

http://apisiar.larioja.org/ultima-hora/id/TU_ID


```
{
“TOTAL”:21,
"DATOS": [
                {
                    "Estacion": "501",
                    "Fecha-Hora": "2016-12-01T12:30:00.000Z"
                },
                {
                     "Estacion": "502",
                     "Fecha-Hora": "2016-12-01T12:30:00.000Z"
                },
                 …
            ]
}
```

RETOS INFORMÁTICOS

- Pequeñas mejoras en la aplicación (hay bastantes detalles):
  - Mejorar las estructuras JSON enviadas, casi todas válidas y bien formadas pero todas mejorables
  - Incluir Versión

http://apisiar.larioja.org/v1/ultima-hora/id/TU_ID

http://apisiar.larioja.org/v2/ultima-hora/id/TU_ID

RETOS INFORMÁTICOS

- Pequeñas mejoras en la aplicación (hay bastantes detalles):
  - Mejorar las estructuras JSON enviadas, casi todas válidas y bien formadas pero todas mejorables
  - Incluir Versión
  - Codificación UTF-8 real

http://apisiar.larioja.org/estaciones-disponibles/id/TU_ID

…
"ESTACION9":{"CODIGO":"509","NOMBRE":"Logro?o"},
…
,"ESTACION14":{"CODIGO":"514","NOMBRE":"CERVERA-CABRET?N"},
…

RETOS INFORMÁTICOS

- Pequeñas mejoras en la aplicación (hay bastantes detalles):
  - Mejorar las estructuras JSON enviadas, casi todas válidas y bien formadas pero todas mejorables
  - Incluir Versión
  - Codificación UTF-8 real
  - Cabecera HTTP que indique “application/json” o “text/plain” (actualmente “text/html”)

RETOS INFORMÁTICOS

- Pequeñas mejoras en la aplicación (hay bastantes detalles):
  - Mejorar las estructuras JSON enviadas, casi todas válidas y bien formadas pero todas mejorables
  - Incluir Versión
  - Codificación UTF-8 real
  - Cabecera HTTP que indique “application/json” o “text/plain” (actualmente “text/html”)
  - Protocolo HTTPS

RETOS INFORMÁTICOS

- Pequeñas mejoras en la aplicación (hay bastantes detalles):
  - Mejorar las estructuras JSON enviadas, casi todas válidas y bien formadas pero todas mejorables
  - Incluir Versión
  - Codificación UTF-8 real
  - Cabecera HTTP que indique “application/json” o “text/plain” (actualmente “text/html”)
  - Protocolo HTTPS
  - Documentación extensiva

RETOS INFORMÁTICOS

- Mejoras de concepto
  - Sistema que haga las consultas más rápidas

```
294  get '/estacion/:numero/tipo/:tipodat/fechain/:fecha1/fechafin/:fecha2/salida/:salida/niveles/:niveles/id/:id' do
295
296      t1 = Time.now
297      fallos= Array.new
298      dat1 = Parsear.new #Creamos el objeto para poder llamar a los métodos de consulta y parseo.
299
300      nombreArchivo = dat1.GeneraNombreArchivo(params[:numero].to_i,params[:tipodat].to_s,params[:fecha1],"0000",params[:fecha2],"233
301      compruebaSiYaExiste = dat1.CompruebaExistenciaArchivo(nombrArchivo)
302
303      if compruebaSiYaExiste
304          send_file(nombreArchivo)
305      else
306          fallos.clear()
307          # COMPROBAR QUE LOS PARAMETROS INTRODUCIDOS SON LOS ESPERABLES
308          fallos= dat1.compruebacampos(params[:numero].to_i,params[:tipodat].to_s,params[:fecha1],params[:fecha2],"sinhora","sinhora"
309          dat = Metodos.new
310
311          # SI HAY ERRORES ENTONCES SE GESTIONA EL ERROR, SINO SE REALIZA LA PETICION
312          if(fallos.size > 0)
313              error=dat1.gestionaerror(1,fallos)
314
315              File.open("Error.txt", "w") do |f1|
316                  f1.puts(error)
317              end
318
319              send_file("Error.txt")
320          else
321              res = dat.metodogeneral4(params[:numero].to_i,params[:tipodat].to_s,params[:fecha1],"0000",params[:fecha2],"2330","todo
322          end
323
324          File.open(nombreArchivo, "w") do |f1|
325              f1.puts(res)
326          end
327
328          send_file(nombreArchivo)
329  end
```

```
294  get '/estacion/:numero/tipo/:tipodat/fechain/:fecha1/fechafin/:fecha2/salida/:salida/niveles/:niveles/id/:id' do
295
296      t1 = Time.now
297      fallos= Array.new
298      dat1 = Parsear.new #Creamos el objeto para poder llamar a los métodos de consulta y parseo.
299
300      nombreArchivo = dat1.GeneraNombreArchivo(params[:numero].to_i,params[:tipodat].to_s,params[:fecha1],"0000",params[:fecha2],"233
301      compruebaSiYaExiste = dat1.CompruebaExistenciaArchivo(nombrArchivo)
302
303      if compruebaSiYaExiste
304          send_file(nombreArchivo)
305      else
306          fallos.clear()
307          # COMPROBAR QUE LOS PARAMETROS INTRODUCIDOS SON LOS ESPERABLES
308          fallos= dat1.compruebacampos(params[:numero].to_i,params[:tipodat].to_s,params[:fecha1],params[:fecha2],"sinhora","sinhora"
309          dat = Metodos.new
310
311          # SI HAY ERRORES ENTONCES SE GESTIONA EL ERROR, SINO SE REALIZA LA PETICION
312          if(fallos.size > 0)
313              error=dat1.gestionaerror(1,fallos)
314
315              File.open("Error.txt", "w") do |f1|
316                  f1.puts(error)
317              end
318
319              send_file("Error.txt")
320          else
321              res = dat.metodogeneral4(params[:numero].to_i,params[:tipodat].to_s,params[:fecha1],"0000",params[:fecha2],"2330","todo
322          end
323
324          File.open(nombreArchivo, "w") do |f1|
325              f1.puts(res)
326          end
327
328          send_file(nombreArchivo)
329  end
```

```ruby
get '/estacion/:numero/tipo/:tipodat/fechain/:fecha1/fechafin/:fecha2/salida/:salida/niveles/:niveles/id/:id' do

    t1 = Time.now
    fallos= Array.new
    dat1 = Parsear.new #Creamos el objeto para poder llamar a los métodos de consulta y parseo.

    nombreArchivo = dat1.GeneraNombreArchivo(params[:numero].to_i,params[:tipodat].to_s,params[:fecha1],"0000",params[:fecha2],"233
    compruebaSiYaExiste = dat1.CompruebaExistenciaArchivo(nombrArchivo)

    if compruebaSiYaExiste
        send_file(nombreArchivo)
    else
        fallos.clear()
        # COMPROBAR QUE LOS PARAMETROS INTRODUCIDOS SON LOS ESPERABLES
        fallos= dat1.compruebacampos(params[:numero].to_i,params[:tipodat].to_s,params[:fecha1],params[:fecha2],"sinhora","sinhora"
        dat = Metodos.new

        # SI HAY ERRORES ENTONCES SE GESTIONA EL ERROR, SINO SE REALIZA LA PETICION
        if(fallos.size > 0)
            error=dat1.gestionaerror(1,fallos)

            File.open("Error.txt", "w") do |f1|
                f1.puts(error)
            end

            send_file("Error.txt")
        else
            res = dat.metodogeneral4(params[:numero].to_i,params[:tipodat].to_s,params[:fecha1],"0000",params[:fecha2],"2330","todo
        end

        File.open(nombreArchivo, "w") do |f1|
            f1.puts(res)
        end

        send_file(nombreArchivo)
end
```

RETOS INFORMÁTICOS

- Mejoras de concepto
  - Sistema que haga las consultas más rápidas
  - Uso del timeStamp

RETOS INFORMÁTICOS

- Mejoras de concepto
  - Sistema que haga las consultas más rápidas
  - Uso del timeStamp

    - Todos los registros de las tablas de datos climáticos tienen un campo fecha que indica el momento en el que se escribe el registro

RETOS INFORMÁTICOS

- Mejoras de concepto
  - Sistema que haga las consultas más rápidas
  - Uso del timeStamp

    - Todos los registros de las tablas de datos climáticos tienen un campo fecha que indica el momento en el que se escribe el registro

    - Gracias a este campo se puede confirmar si el campo ha sufrido o no cambios desde la última consulta

RETOS INFORMÁTICOS

- Mejoras de concepto
  - Sistema que haga las consultas más rápidas
  - Uso del timeStamp
  - Mantenibilidad del código, migración a GRAPE

RETOS INFORMÁTICOS

- Mejoras de concepto
  - Sistema que haga las consultas más rápidas
  - Uso del timeStamp
  - Mantenibilidad del código, migración a GRAPE

https://github.com/ruby-grape/grape

RETOS INFORMÁTICOS

- Mejoras de concepto
  - Sistema que haga las consultas más rápidas
  - Uso del timeStamp
  - Mantenibilidad del código, migración a GRAPE

What is Grape?
Grape is a REST-like API framework for Ruby. It's designed to run on Rack or complement existing web application frameworks such as Rails and Sinatra by providing a simple DSL to easily develop RESTful APIs. It has built-in support for common conventions, including multiple formats, subdomain/prefix restriction, content negotiation, versioning and much more.

RETOS INFORMÁTICOS

- Mejoras de concepto
  - Sistema que haga las consultas más rápidas
  - Uso del timeStamp
  - Mantenibilidad del código, migración a GRAPE
  - Disponibilidad de los datos

RETOS INFORMÁTICOS

- Mejoras de concepto
  - Sistema que haga las consultas más rápidas
  - Uso del timeStamp
  - Mantenibilidad del código, migración a GRAPE
  - Disponibilidad de los datos

¿es necesario poner a disposición del usuario TODOS los datos semihorarios históricos?

RETOS INFORMÁTICOS

- Mejoras de concepto
  - Sistema que haga las consultas más rápidas
  - Uso del timeStamp
  - Mantenibilidad del código, migración a GRAPE
  - Disponibilidad de los datos

¿es necesario poner a disposición del usuario TODOS los datos semihorarios históricos?

¿Y los diarios?

RETOS INFORMÁTICOS

- Mejoras de concepto
  - Sistema que haga las consultas más rápidas
  - Uso del timeStamp
  - Mantenibilidad del código, migración a GRAPE
  - Disponibilidad de los datos

¿es necesario poner a disposición del usuario TODOS los datos semihorarios históricos?

¿Y los diarios?

¿Datos semanales, mensuales? ¿otra información derivada como índices climáticos, riesgo de enfermedades, etc?

RETOS INFORMÁTICOS

- Mejoras de concepto
  - Sistema que haga las consultas más rápidas
  - Uso del timeStamp
  - Mantenibilidad del código, migración a GRAPE
  - Disponibilidad de los datos
  - Trazabilidad de los datos una vez descargados

RETOS INFORMÁTICOS

- Mejoras de concepto
  - Sistema que haga las consultas más rápidas
  - Uso del timeStamp
  - Mantenibilidad del código, migración a GRAPE
  - Disponibilidad de los datos
  - Trazabilidad de los datos una vez descargados

¿?

RETOS AGRARIOS de apisiar.larioja.org

- Manejo del resultado de la validación

RETOS AGRARIOS de apisiar.larioja.org

- Manejo del resultado de la validación
  - Si se ofrece dato con nivel => el usuario es responsable
  - Si se ofrece dato sin nivel => ¿Qué hacer?

RETOS AGRARIOS de apisiar.larioja.org

- Manejo del resultado de la validación
- Incorporar información propiedad del dato

RETOS AGRARIOS de apisiar.larioja.org

- Manejo del resultado de la validación
- Incorporar información propiedad del dato
- Distribuir datos con licencia que permita
  - Reutilización datos dándole valor añadido
  - Reconocimiento fuente original del dato

RETOS AGRARIOS de apisiar.larioja.org

- Manejo del resultado de la validación
- Incorporar información propiedad del dato
- Distribuir datos con licencia que permita
  - Reutilización datos dándole valor añadido
  - Reconocimiento fuente original del dato
- Adaptar la salida de datos a estándares distribución información p. ej.: INSPIRE

MUCHAS GRACIAS

Servicio de Información Agroclimática de La Rioja (SIAR)
Servicio de Producción Agraria y Laboratorio Regional – Consejería de Agricultura - Gobierno de La Rioja

Finca La Grajera
Ctra. LO-20 - salida 13
Autovía del Camino de Santiago
26071 – Logroño (La Rioja)

siar.cida@larioja.org
941 29 18 34