



¿Quienes somos?

► Luciano Rossi

► Sebastián Marró



¡Bienvenidos!

Tryton

# Software Libre

*"Seamos libres lo demas no importa nada"*

- ▶ usar
- ▶ estudiar
- ▶ distribuir
- ▶ mejorar







Plataforma de desarrollo de aplicaciones  
de alto nivel Arquitectura de 3 Capas.

- \* GPLv3
- \* Cliente PyGtk / Cliente Web (sao)  
Ligero (No tiene logica de negocios)
- \* Servidor Python
- \* Postgresql / SQLite / MySQL

A background image showing a person from the back, wearing a dark blue shirt with the word 'Tryton' printed on the back. The person is standing outdoors, and their hands are visible, holding a small object. The image is dark and serves as a backdrop for the text.

Objetivos de Tryton

Consistencia

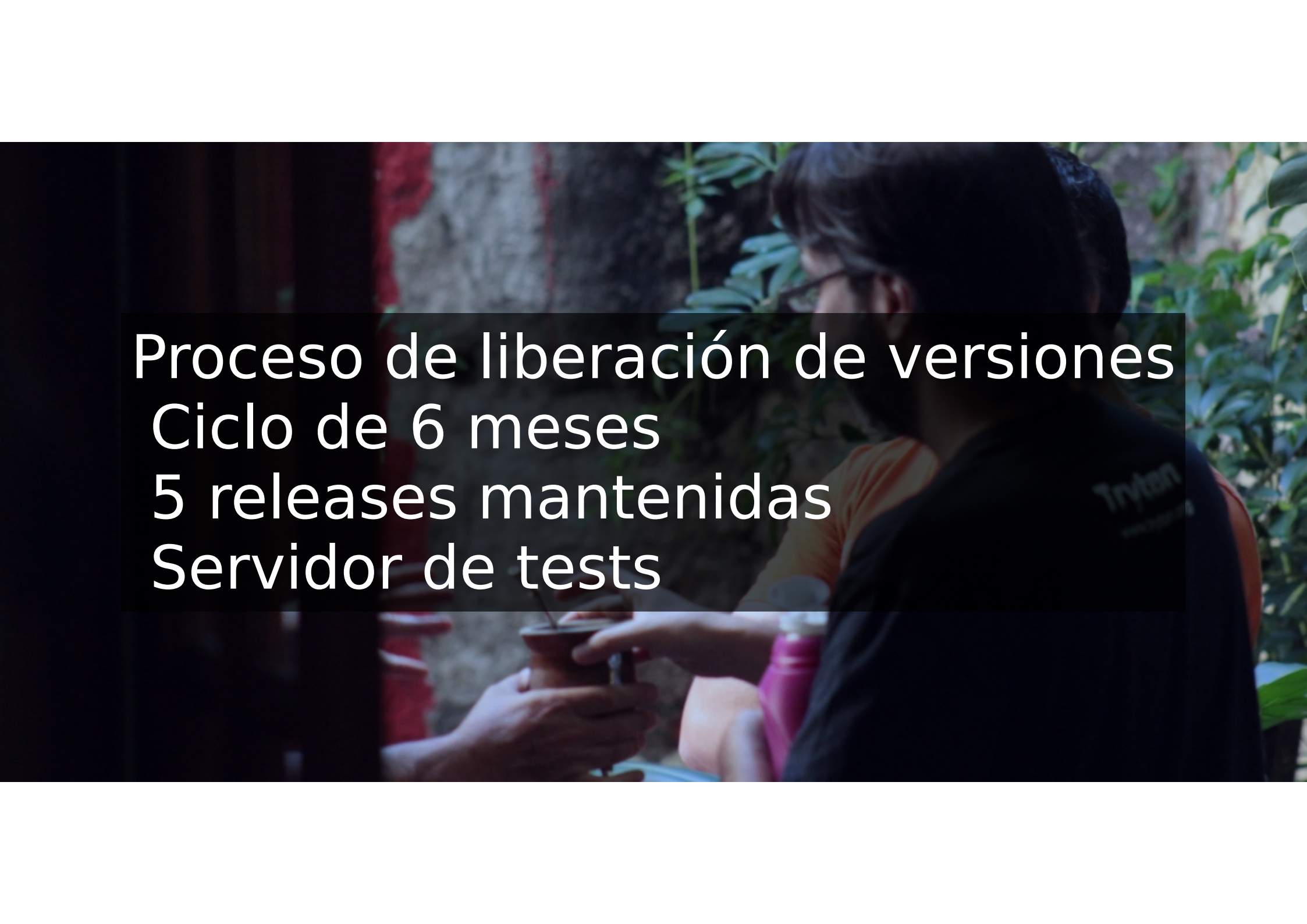
Migración

Modularidad

Enfoque “Framework especializado”

Paquetes Python → pip

Code review



Proceso de liberación de versiones  
Ciclo de 6 meses  
5 releases mantenidas  
Servidor de tests



A background image of a person with dark hair and glasses, wearing a dark jacket, standing in a garden with green foliage. The image is partially obscured by a dark grey overlay on the left and bottom.

# Servidor Python

## ORM

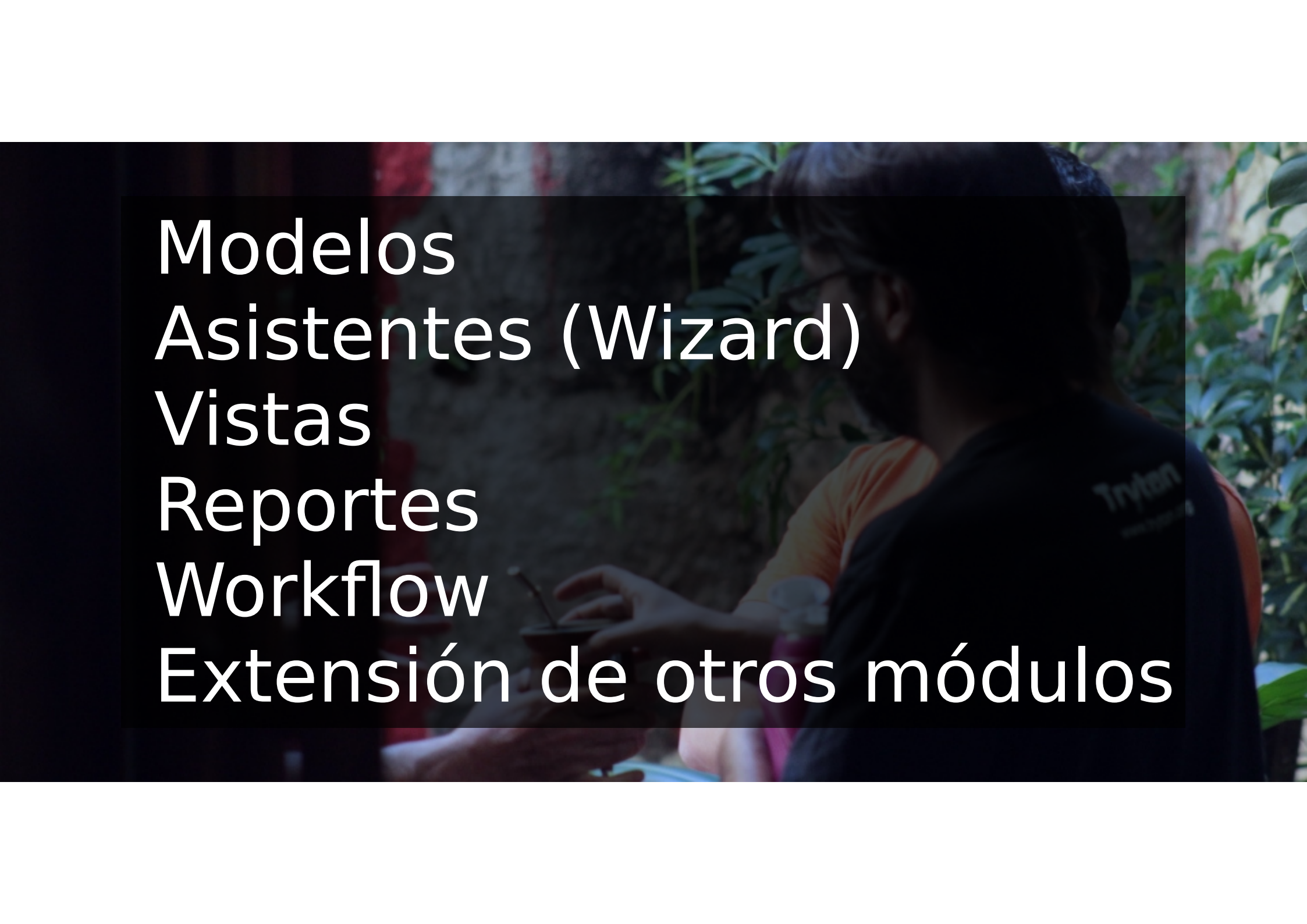
Creación de esquema

Migración automática

Workflows

Reportes (relatorio)

Multi-lenguajes



Modelos  
Asistentes (Wizard)  
Vistas  
Reportes  
Workflow  
Extensión de otros módulos



A background image showing a person from the side, wearing glasses and a dark shirt, holding a traditional mate cup with a bombilla. The person is outdoors, with green foliage visible in the background. The image is dimmed to serve as a background for the text.

# Modelos

Modelos

Campos

Valores por defecto

Search – Create – Write – Delete

Browse

Vistas

Estados

```
1 from trytond.model import ModelSQL
2
3 class Event (ModelSQL):
4     'Event'
5     __name__ = 'calendar.event'
```

```
1 class Event (ModelSQL, ModelView):
2     'Event '
3     __name__ = 'calendar.event'
4
5     calendar = fields.Many2One('calendar.calendar', 'Calendar',
6                                required=True)
7     summary = fields.Char('Summary')
8     description = fields.Text('Description')
9     dtstart = fields.DateTime('Start Date', required=True)
10    classification = fields.Selection([
11        ('public', 'Public'),
12        ('private', 'Private'),
13    ], 'Classification', required=True)
```



```
1 class Event (ModelSQL, ModelView):
2
3     classification = fields.Selection([
4         ('public', 'Public'),
5         ('private', 'Private'),
6     ], 'Classification', required=True)
7
8     @staticmethod
9     def default_classification():
10         return 'public'
```

```
1 calendar = Calendar.search([
2     ('owner.email', '=', 'juan@mycompany.com'),
3     ])
4
5 event = Event.create({
6     'summary': 'Reunion',
7     'calendar': calendar.id,
8     })
9
10 Event.write(event.id, {
11     'description': 'Planificar charla PyConAr',
12     })
13
14 Event.delete(event.id)
```

```
1 >>> event = Event(event_id)
2
3 >>> print event.dtstart
4 datetime.date(2012, 11, 7)
5
6 >>> for attendee in event.attendees:
7     ...     print attendee.name
8 Han Solo
9 Luke Skywalker
```



```
1 <record model="ir.ui.view" id="event_view_tree">
2   <field name="model">calendar.event</field>
3   <field name="type">tree</field>
4   <field name="arch" type="xml">
5     <![CDATA[
6       <tree string="Events">
7         <field name="calendar"/>
8         <field name="summary"/>
9         <field name="dtstart"/>
10        <field name="dtend"/>
11      </tree>
12    ]]>
13   </field>
14 </record>
```

```
1 <record model="ir.ui.view" id="event_view_form">
2   <field name="model">calendar.event</field>
3   <field name="type">form</field>
4   <field name="arch" type="xml">
5     <![CDATA[
6       <form string="Event">
7         <label name="summary"/>
8         <field name="summary"/>
9         <separator name="description" colspan="4"/>
10        <field name="description" colspan="4"/>
11      </form>
12    ]]>
13   </field>
14 </record>
```

```
1 from trytond.pyson import If, Bool, Eval
2
3 class Event(ModelSQL, ModelView):
4
5     attendees = fields.One2Many('calendar.event.attendee', 'event',
6                                 'Attendees')
7     organizer = fields.Char('Organizer', states={
8         'required': If(Bool(Eval('attendees')), ~Eval('parent'), False),
9         }, depends=['attendees'])
```



A background image showing a person from the side, wearing a dark blue shirt with the 'Tryton' logo on the sleeve. They are holding a small dark cup with a metal straw. The background is slightly blurred, showing some green foliage and a wall with red markings.

Proteus

Librería para acceder a los modelos  
de Tryton como un cliente

CLI Python

## 1 Creando un Party

2 *no no no no no no no no no no no no no no no no*

3

4 Primero instanciamos un nuevo Party:

5

6 >>> Party = Model.get('party.party')

7 >>> party = Party()

8 >>> party.id < 0

9 True

10

11 Llenamos los campos:

12

13 >>> party.name = 'ham'

14

15 Grabamos la instancia en el servidor:

16

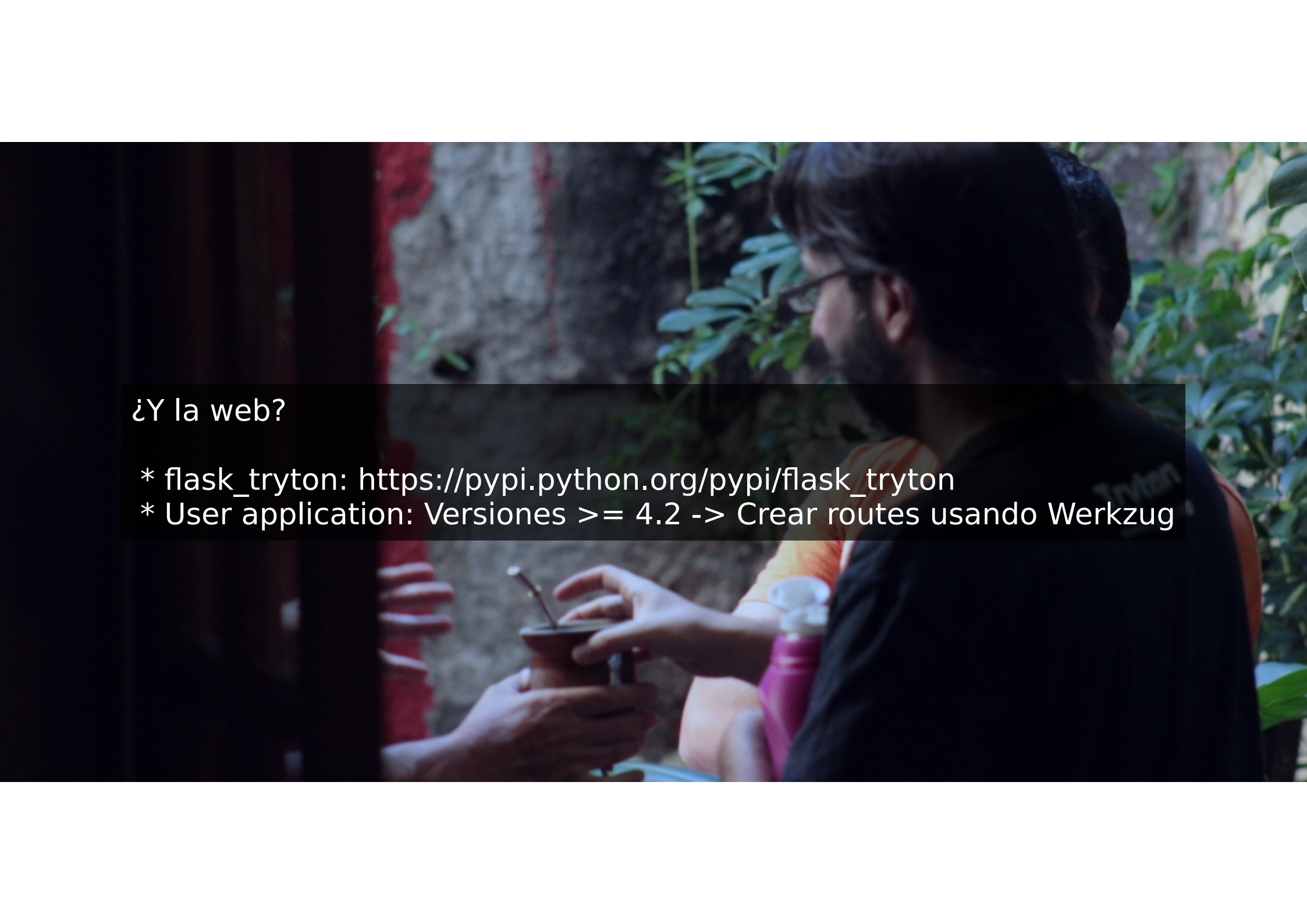
17 >>> party.save()

18 >>> party.name

19 u'ham'

20 >>> party.id > 0

21 True



¿Y la web?

- \* flask\_tryton: [https://pypi.python.org/pypi/flask\\_tryton](https://pypi.python.org/pypi/flask_tryton)

- \* User application: Versiones  $\geq 4.2$  -> Crear routes usando Werkzeug



```
@app.route('/rest/v1/invoices/<record("account.invoice"):invoice>',
           methods=['POST'])
@auth.login_required
@tryton.transaction()
def invoice_reporte(invoice):
    "Crear Reporte"

    InvoiceReport = tryton.pool.get('account.invoice', type='report')
    try:
        type_, file_data, print_, name = InvoiceReport.execute([invoice.id], {})
    except:
        abort(404, u'Nonexistent invoice')
    file_data_encode = base64.b64encode(str(file_data))
    data = {
        'invoice_id': invoice.id,
        'state': invoice.state,
        'report_name': name,
        'report_data': file_data_encode,
        'report_type': mimetypes.types_map['.'+type_], # pasar el mimetype
        'report_ext': type_,
    }

    return jsonify({'data': data}), 200
```

Michael Scott



11/25/2016



**Duration**

**Work**

**Description**

02:00 Marketing

03:00 Secretary

Add

Crear una aplicación agregando routes en tu módulo usando Werkzeug.  
Un ejemplo que podemos encontrar:  
<https://addons.mozilla.org/en-US/firefox/addon/tryton-chronos/>

A background image showing a person from the back, wearing a dark blue shirt with the 'Tryton' logo and 'www.tryton.org' on the sleeve. They are standing outdoors near some green plants. A semi-transparent dark rectangle is overlaid on the left side of the image, containing white text.

# Contribuir en Tryton

## Testear

## Realizar codereview

## Subir parches

A person with dark hair and glasses, wearing a dark blue t-shirt with the word 'Tryton' in orange on the back, is shown from the side. They are holding a small object in their hands. The background is slightly blurred, showing some greenery and a wall.

¿Dónde consultar?

Web: <http://www.tryton.org/>

IRC: #tryton y #tryton-es en [irc.freenode.net](http://irc.freenode.net)

tryton-ar en Google Groups



The logo for Tryton Argentina is a stylized, multi-faceted geometric shape. It features a central white vertical element, with light blue and dark blue triangular facets extending from it. The overall shape is reminiscent of a modern building or a crystal structure.

## Tryton Argentina

<https://github.com/tryton-ar>

account\_ar: Plan de cuentas de para empresas Argentinas.

account\_coop\_ar: Plan de cuentas para cooperativas de Trabajo.

party\_ar: Integracion del padron AFIP.

account\_invoice ar: Facturación integrada con ws de la AFIP.

account\_check\_ar: Manejo de cheques

account\_voucher\_ar: Comprobantes de pago.

account\_retencion\_ar: Retenciones

bank\_ar: Incorpora los datos de bancos de Argentina.

cooperative\_ar: Gestion de Cooperativa (Recibos, Vacaciones, Socios, etc)

subdiario: Reportes que generan el subdiario

citi\_afip: Informativo de Compras y Ventas

**Got a question?**

**ASK  
ALF**

