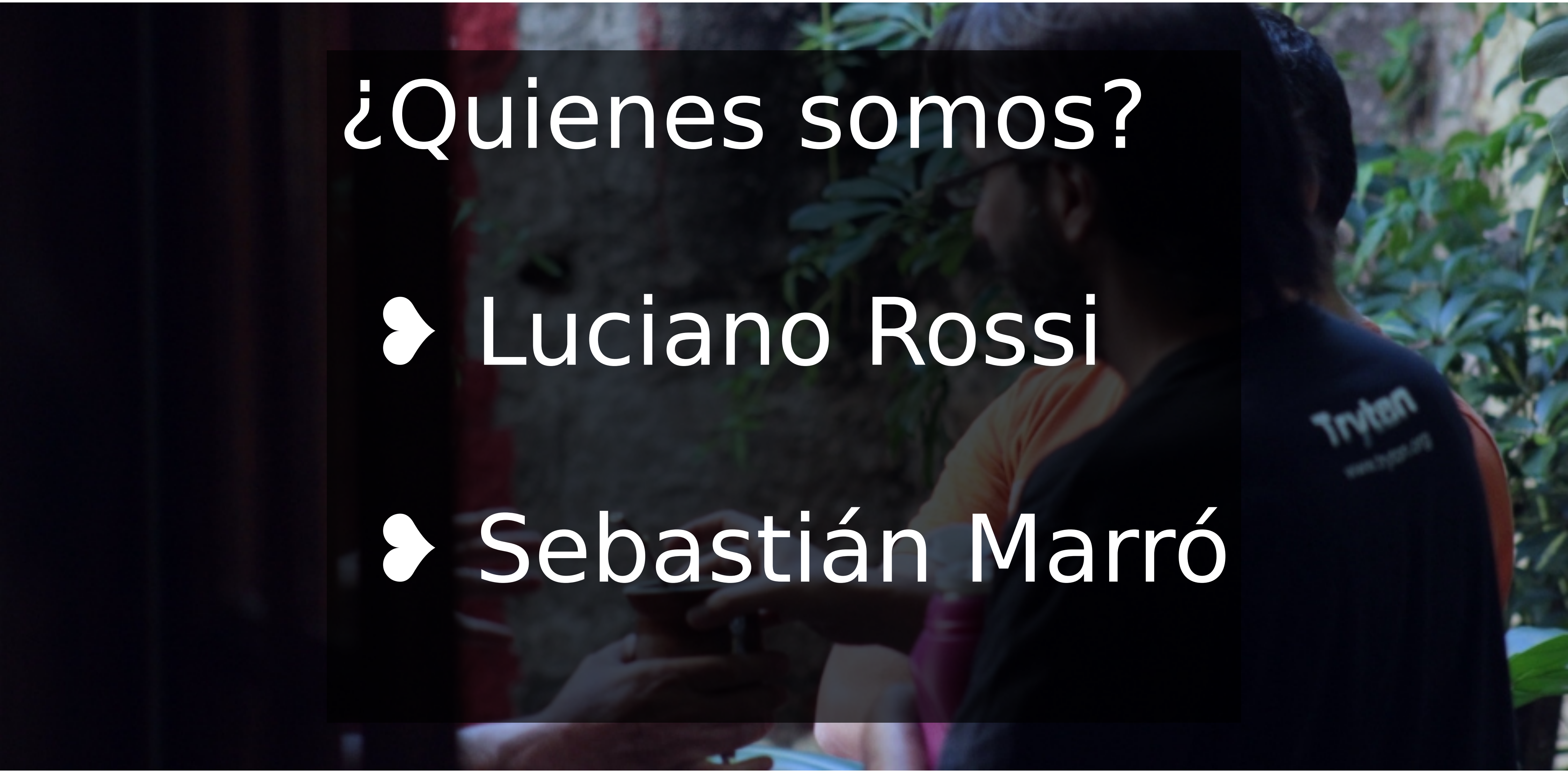


¿Quienes somos?

- Luciano Rossi
- Sebastián Marró

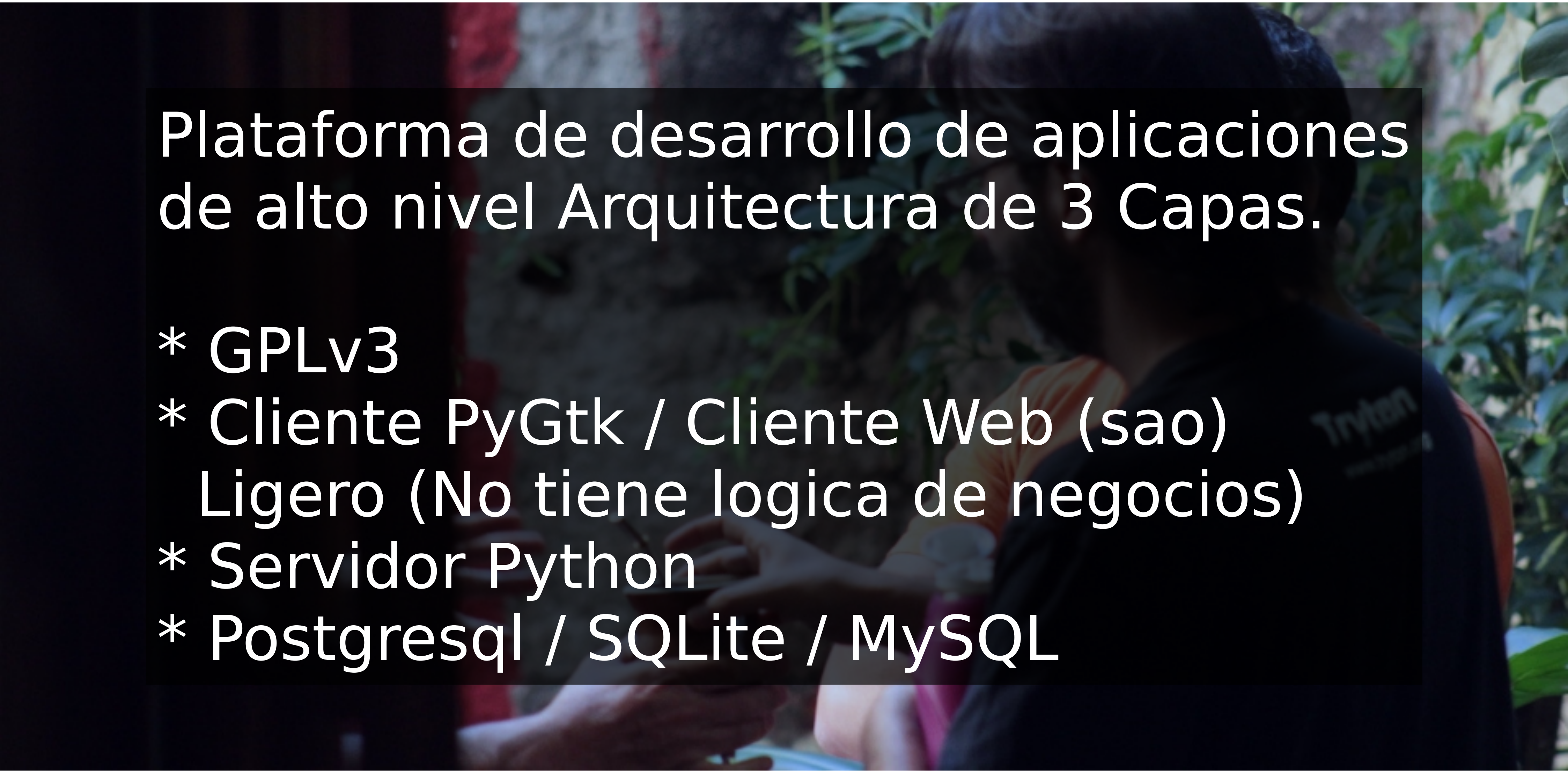
¡Bienvenidos!

Tryton

Software Libre

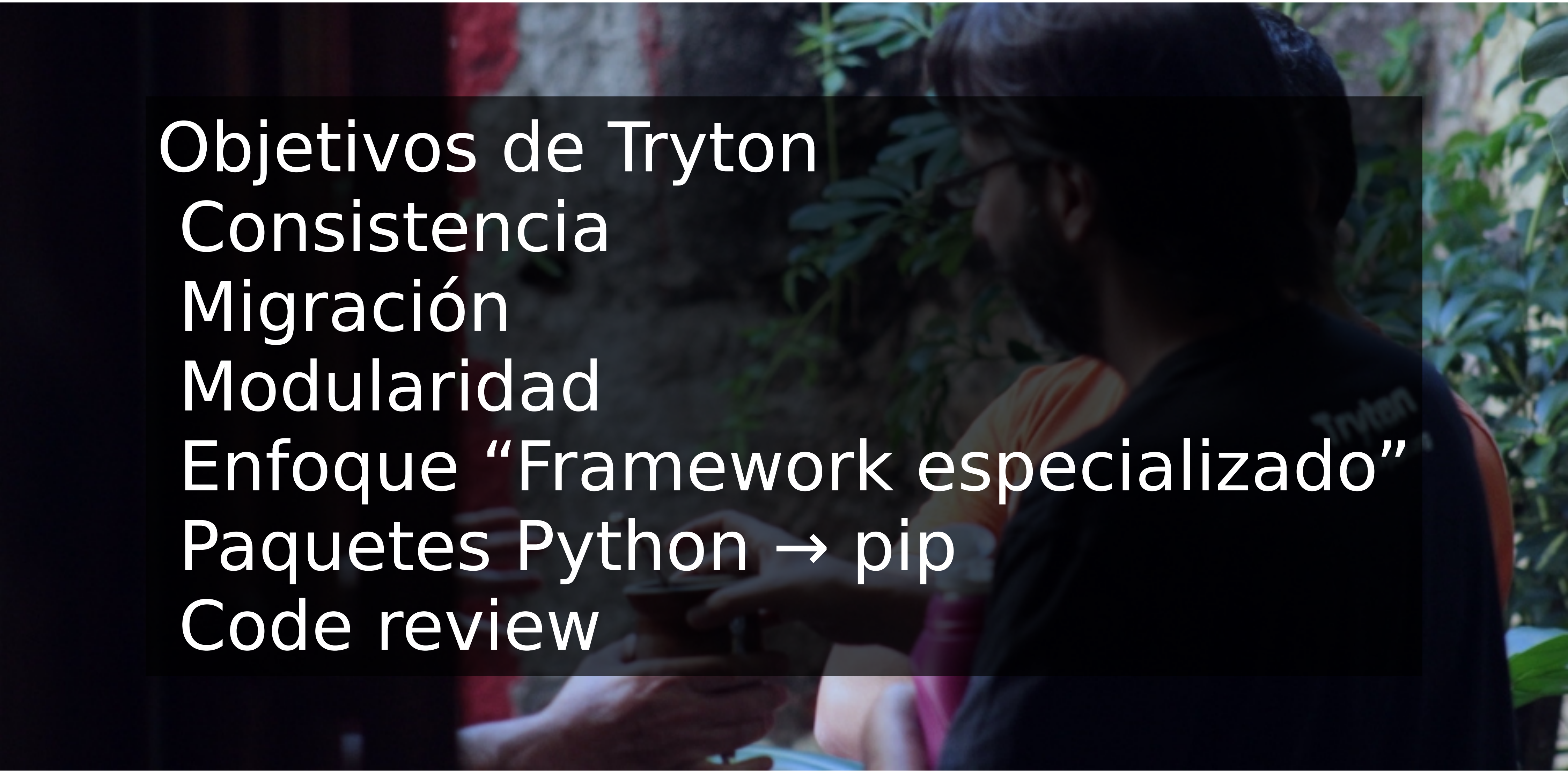
"*Seamos libres lo demas no importa nada*"

- usar
- estudiar
- distribuir
- mejorar



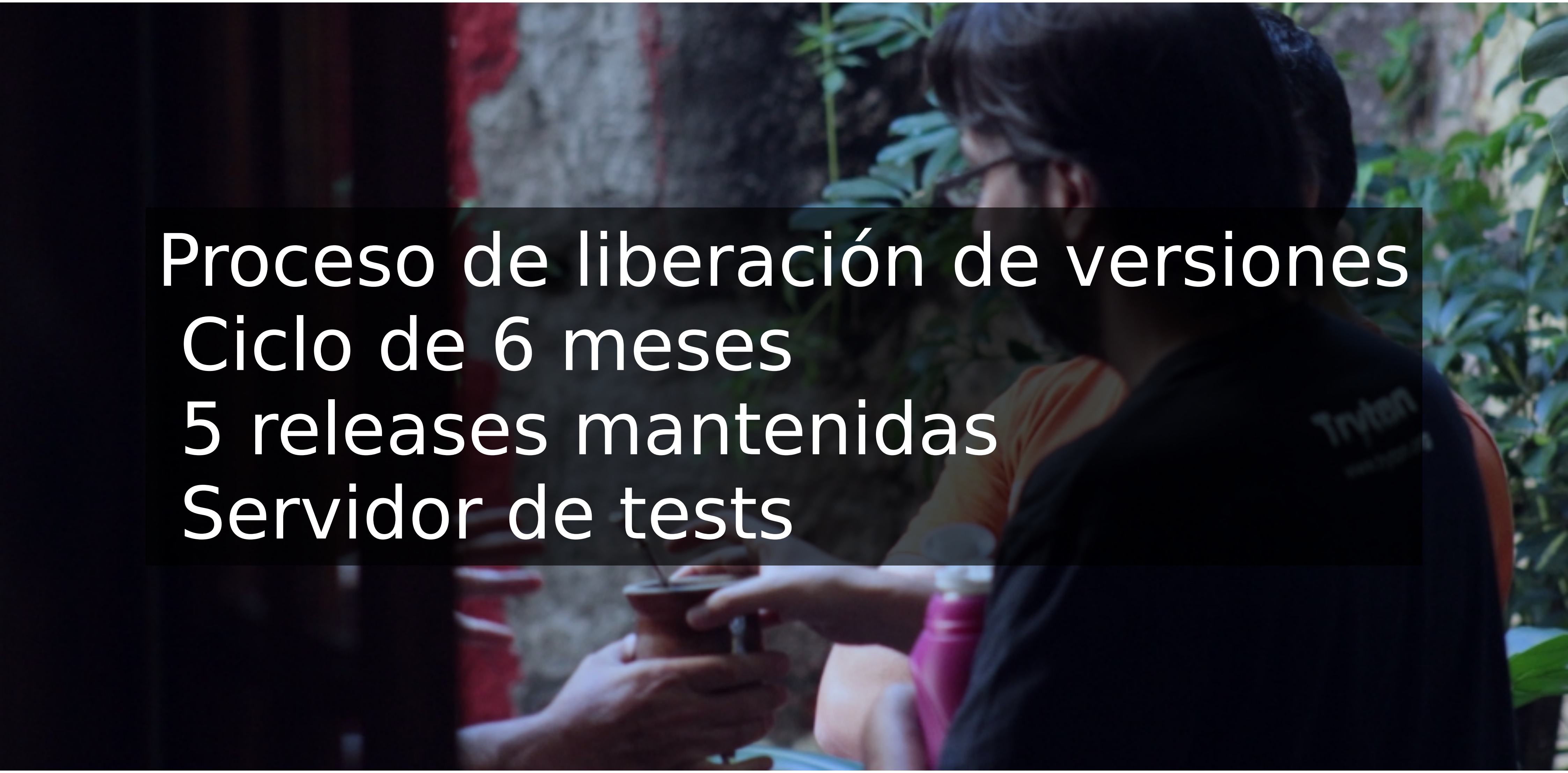
Plataforma de desarrollo de aplicaciones de alto nivel Arquitectura de 3 Capas.

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



Objetivos de Tryton

- Consistencia
- Migración
- Modularidad
- Enfoque “Framework especializado”
- Paquetes Python → pip
- Code review



Proceso de liberación de versiones

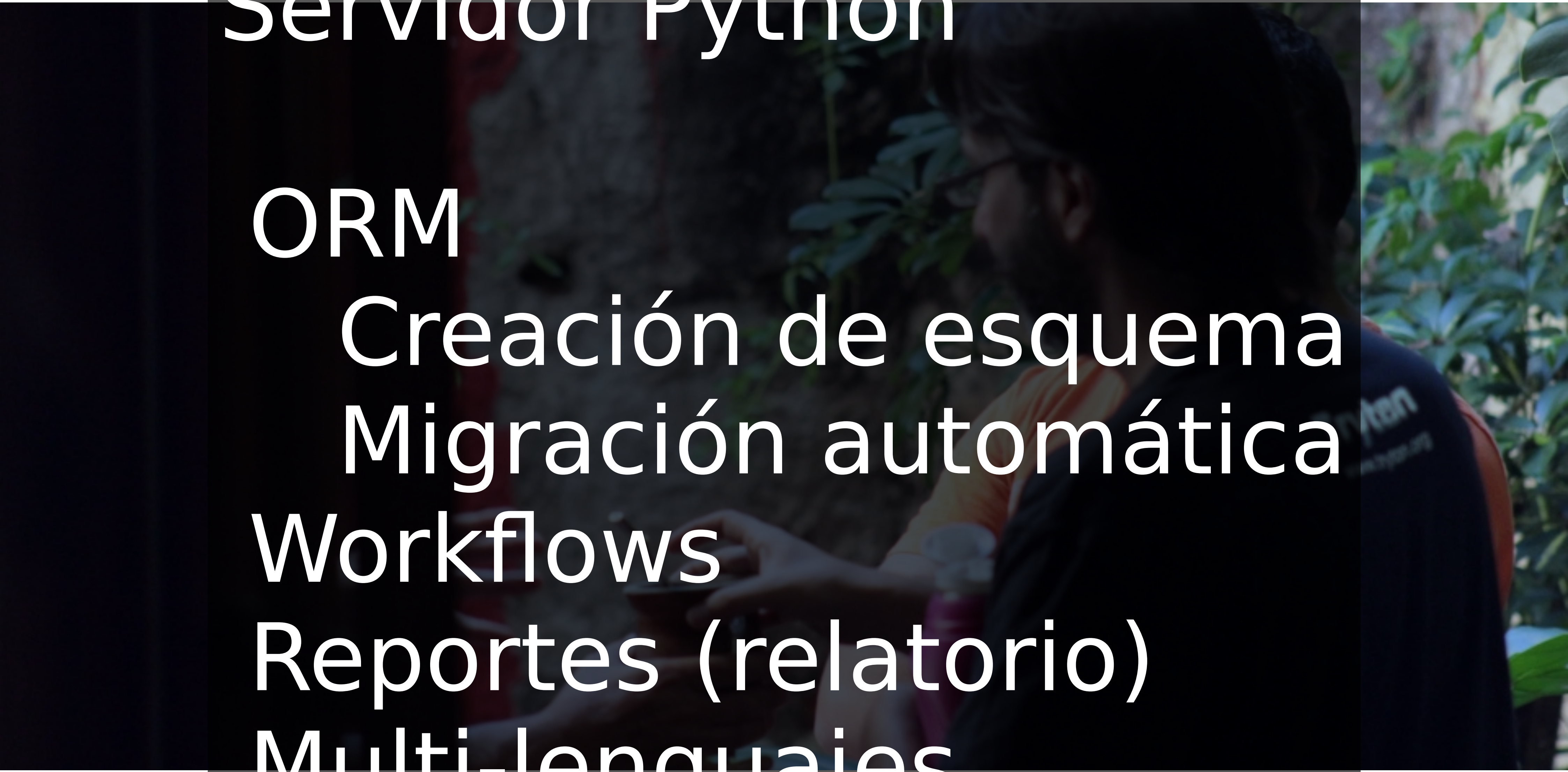
Ciclo de 6 meses

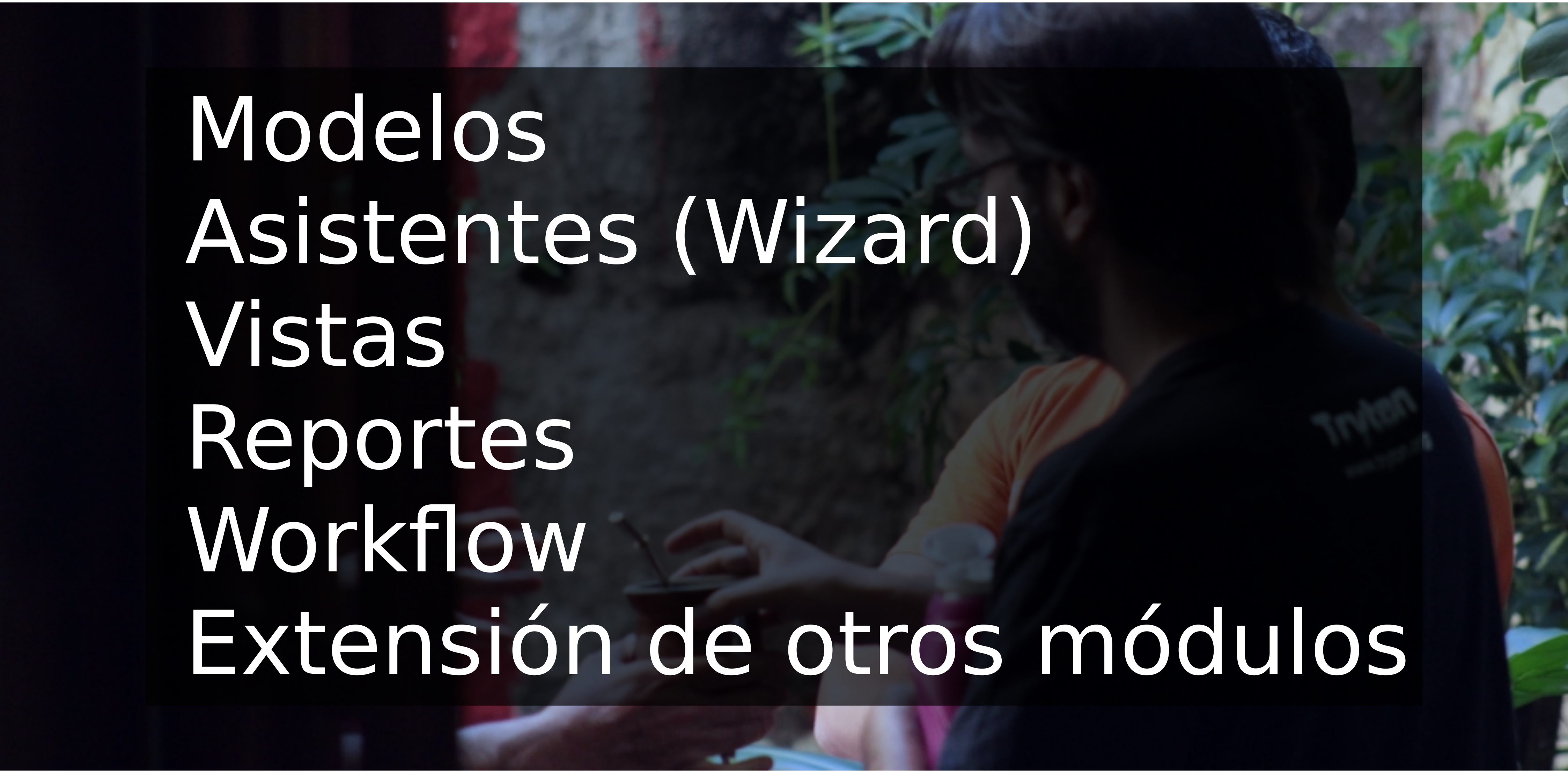
5 releases mantenidas

Servidor de tests

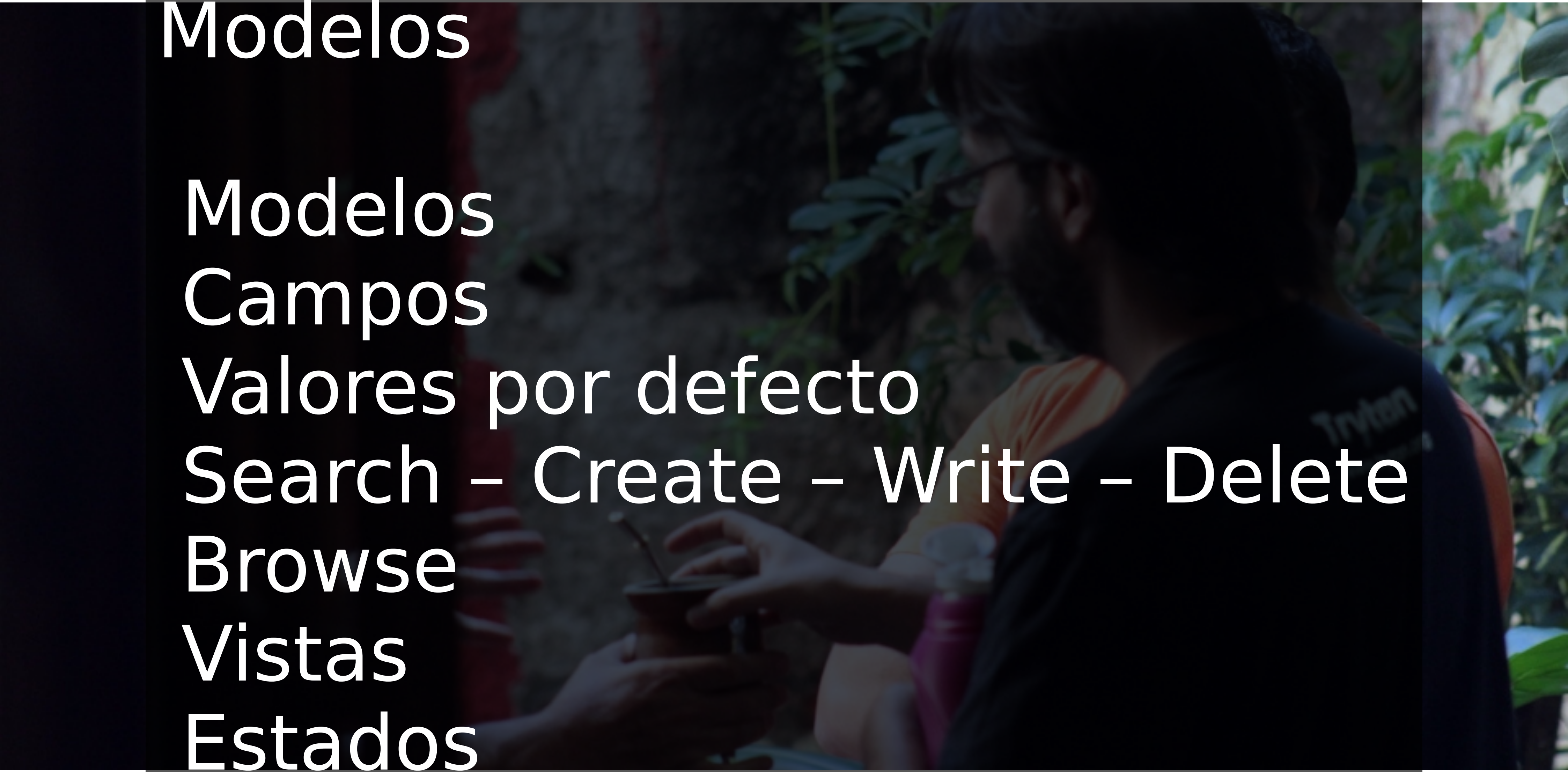
Servidor Python

- ORM
  - Creación de esquema
  - Migración automática
- Workflows
- Reportes (relatorio)
- Multi-lenguajes





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

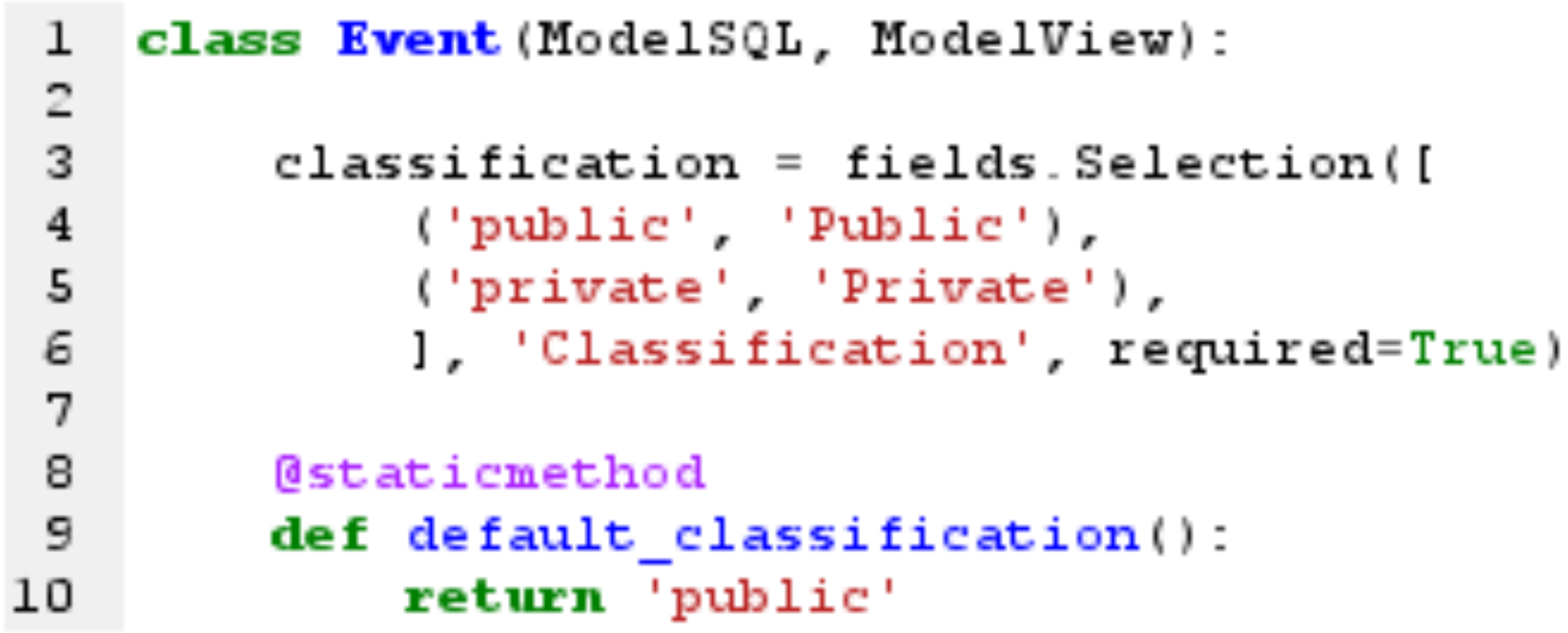


Modelos

- Modelos
- Campos
- Valores por defecto
- Search – Create – Write – Delete
- Browse
- Vistas
- Estados

```python
from trytond.model import ModelSQL

class Event(ModelSQL):
    'Event'
    __name__ = 'calendar.event'
```

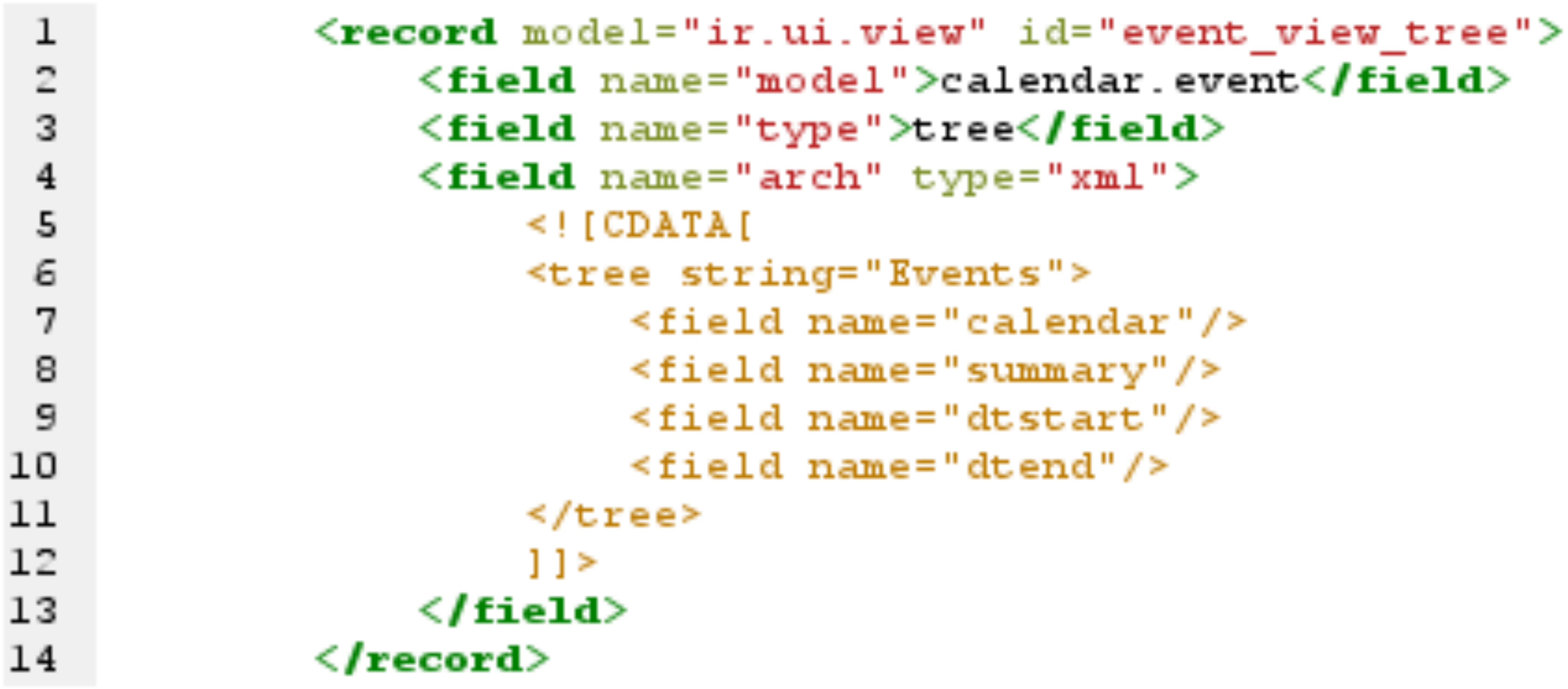
```python
class Event(ModelSQL, ModelView):
    'Event'
    __name__ = 'calendar.event'

    calendar = fields.Many2One('calendar.calendar', 'Calendar',
            required=True)
    summary = fields.Char('Summary')
    description = fields.Text('Description')
    dtstart = fields.DateTime('Start Date', required=True)
    classification = fields.Selection([
        ('public', 'Public'),
        ('private', 'Private'),
        ], 'Classification', required=True)
```



```python
class Event(ModelSQL, ModelView):

    classification = fields.Selection([
        ('public', 'Public'),
        ('private', 'Private'),
        ], 'Classification', required=True)

    @staticmethod
    def default_classification():
        return 'public'
```

```python
calendar = Calendar.search([
    ('owner.email', '=', 'juan@mycompany.com'),
    ])

event = Event.create({
    'summary': 'Reunion',
    'calendar': calendar.id,
    })

Event.write(event.id, {
    'description': 'Planificar charla PyConAr',
    })

Event.delete(event.id)
```

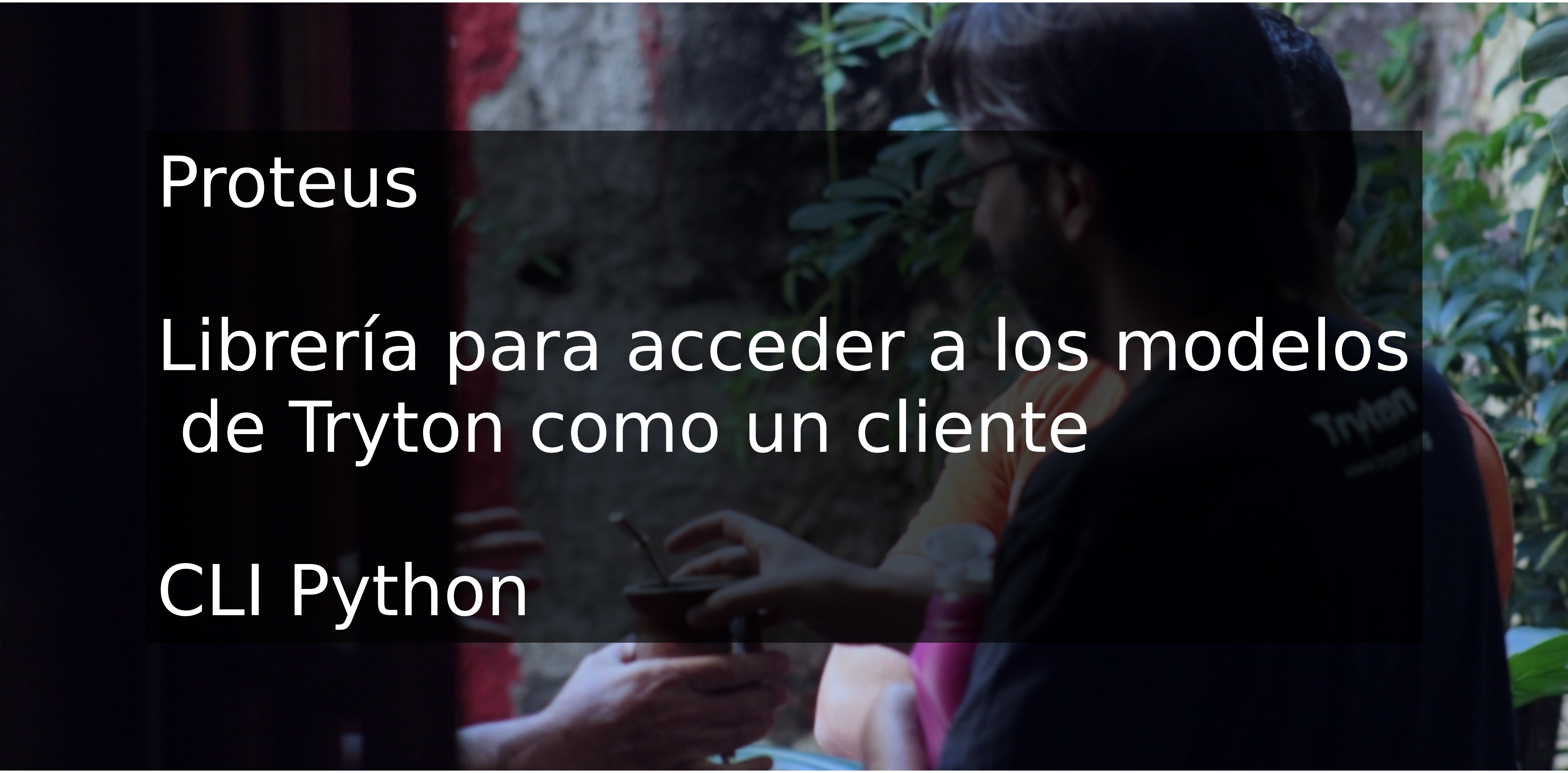
```
>>> event = Event(event_id)

>>> print event.dtstart
datetime.date(2012, 11, 7)

>>> for attendee in event.attendees:
...     print attendee.name
Han Solo
Luke Skywalker
```



```xml
        <record model="ir.ui.view" id="event_view_tree">
            <field name="model">calendar.event</field>
            <field name="type">tree</field>
            <field name="arch" type="xml">
                <![CDATA[
                <tree string="Events">
                    <field name="calendar"/>
                    <field name="summary"/>
                    <field name="dtstart"/>
                    <field name="dtend"/>
                </tree>
                ]]>
            </field>
        </record>
```

```xml
<record model="ir.ui.view" id="event_view_form">
    <field name="model">calendar.event</field>
    <field name="type">form</field>
    <field name="arch" type="xml">
        <![CDATA[
        <form string="Event">
            <label name="summary"/>
            <field name="summary"/>
            <separator name="description" colspan="4"/>
            <field name="description" colspan="4"/>
        </form>
        ]]>
    </field>
</record>
```

```python
from trytond.pyson import If, Bool, Eval

class Event(ModelSQL, ModelView):

    attendees = fields.One2Many('calendar.event.attendee', 'event',
            'Attendees')
    organizer = fields.Char('Organizer', states={
            'required': If(Bool(Eval('attendees')), ~Eval('parent'), False),
            }, depends=['attendees'])
```



Proteus

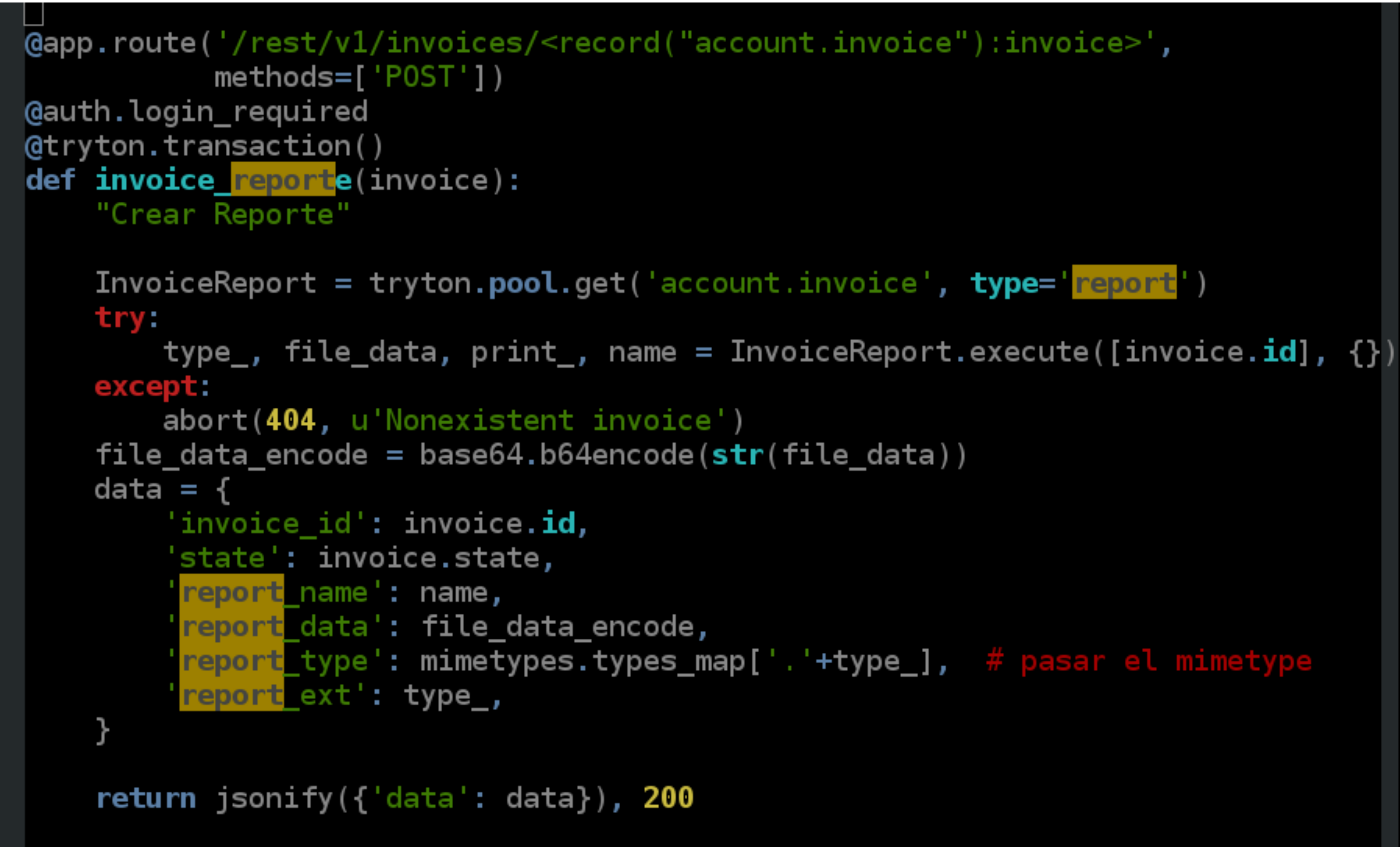
Librería para acceder a los modelos
 de Tryton como un cliente

CLI Python

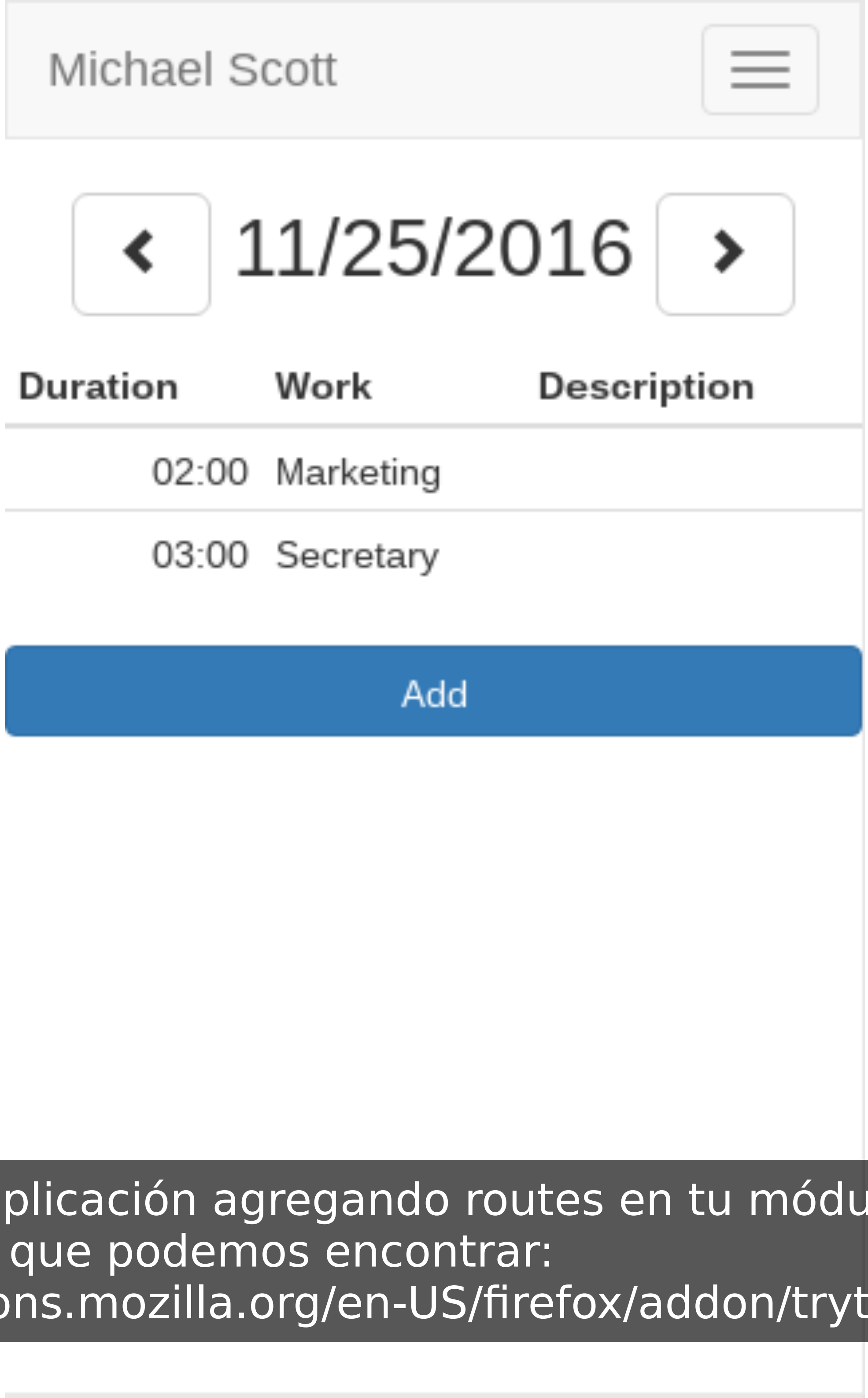
```
Creando un Party
~~~~~~~~~~~~~~~~

Primero instanciamos un nuevo Party:

    >>> Party = Model.get('party.party')
    >>> party = Party()
    >>> party.id < 0
    True

Llenamos los campos:

    >>> party.name = 'ham'

Grabamos la instancia en el servidor:

    >>> party.save()
    >>> party.name
    u'ham'
    >>> party.id > 0
    True
```

¿Y la web?

* flask_tryton: https://pypi.python.org/pypi/flask_tryton
* User application: Versiones >= 4.2 -> Crear routes usando Werkzug

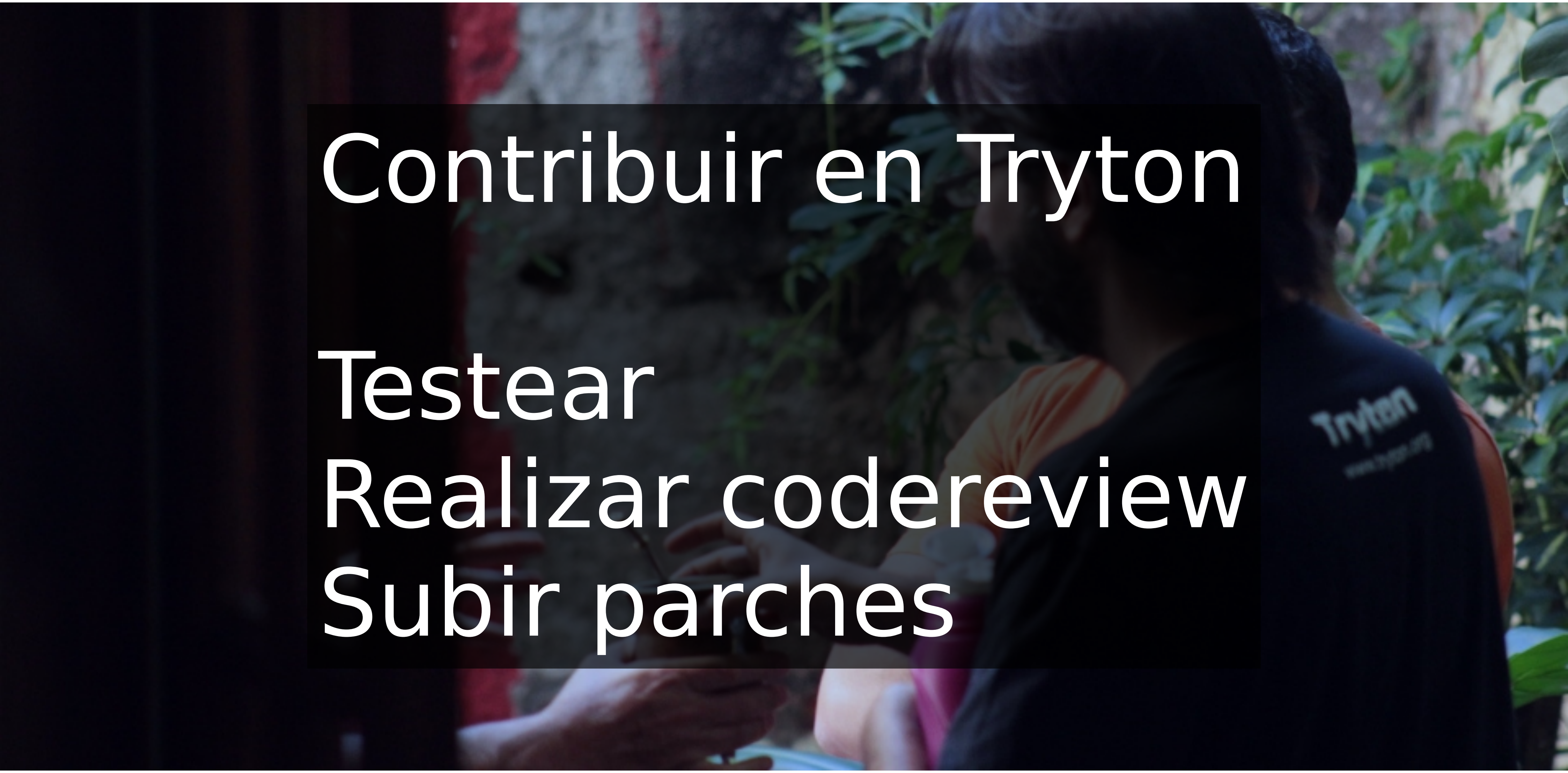


```python
@app.route('/rest/v1/invoices/<record("account.invoice"):invoice>',
           methods=['POST'])
@auth.login_required
@tryton.transaction()
def invoice_reporte(invoice):
    "Crear Reporte"

    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
```



Duration	Work	Description
02:00	Marketing	
03:00	Secretary	

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



Contribuir en Tryton

Testear
Realizar codereview
Subir parches

¿Donde consultar?

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

IRC: #tryton y #tryton-es en irc.freenode.net

tryton-ar en Google Groups

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