

Localizar En Windows e GNU/Linux

Xogando coa hora e coas chamadas a métodos dende distintos puntos do aplicativo

Odoo almacena os "time" en horario UTC

Definimos dous campos, un para traballar coa data e o outro coa data e a hora:

Como vemos utilizamos funcións lambda (anónimas) para inicializar os valores dos campos, **data** co día de hoxe (usando o método **today** de Date) e **data_hora** co día de hoxe e a hora actual (usando o método **now** de Datetime).

```
data = fields.Date(string="Data", default=lambda self: fields.Date.today())
data_hora = fields.Datetime(string="Data e Hora", default=lambda self: fields.Date
```

Como podemos ver, nos marcos vermellos, da seguinte imaxe. Odoo almacena (a hora), por exemplo nos campos Datetime, en horario **UTC**.

Cando visualizamos eses campos dende o aplicativo, Odoo calcula a hora a mostrar tendo en conta a configuración de "timezone" do usuario.

Título	A Descrición	Alto en centímetros	Longo en centímetros	Ancho en centímetros	Volume	Peso en Kg.s	Densidade	¿Autorizado?	Sexo	Data e Hora	Mes castelán	Mes Galego	Hora UTC	Hora Timezone do Usuario	Hora Actual
<input type="checkbox"/> Rexistro 1	Descrición Rexistro 1	1	1	1	1,00	1,00	100,00	<input checked="" type="checkbox"/>	Home	09/04/2020 14:59:43	abril	Abril	12:59:43	14:59:43	12:17:14
<input type="checkbox"/> Rexistro 10	Descrición Rexistro 10	7	4	4	112,00	4,00	3,57	<input checked="" type="checkbox"/>	Muller	14/01/2020 14:59:43	enero	Xaneiro	13:59:43	14:59:43	10:22:08
<input type="checkbox"/> Rexistro 11	Descrición Rexistro 11	7	4	4	112,00	4,00	3,57	<input type="checkbox"/>	Home	09/04/2020 14:59:43	abril	Abril	12:59:43	14:59:43	12:48:41
<input type="checkbox"/> Rexistro 12	Descrición Rexistro 12	4	4	4	64,00	4,00	6,25	<input type="checkbox"/>	Home	09/04/2020 14:59:43	abril	Abril	12:59:43	14:59:43	12:17:14
<input type="checkbox"/> Rexistro 2	Descrición Rexistro 2	2	2	2	8,00	2,00	25,00	<input checked="" type="checkbox"/>	Home	09/04/2020 14:59:43	abril	Abril	12:59:43	14:59:43	12:17:14
<input type="checkbox"/> Rexistro 3	Descrición Rexistro 3	3	3	3	27,00	3,00	11,11	<input checked="" type="checkbox"/>	Home	09/04/2020 14:59:43	abril	Abril	12:59:43	14:59:43	12:17:14
<input type="checkbox"/> Rexistro 4	Descrición Rexistro 4	4	4	4	64,00	4,00	6,25	<input checked="" type="checkbox"/>	Muller	09/04/2020 14:59:43	abril	Abril	12:59:43	14:59:43	12:17:14
<input type="checkbox"/> Rexistro 5	Descrición Rexistro 5	4	4	4	64,00	4,00	6,25	<input type="checkbox"/>	Muller	09/04/2020 14:59:43	abril	Abril	12:59:43	14:59:43	12:44:28
<input type="checkbox"/> Rexistro 6	Descrición Rexistro 6	4	4	4	64,00	4,00	6,25	<input type="checkbox"/>	Muller	09/04/2020 14:59:43	abril	Abril	12:59:43	14:59:43	12:17:14
<input type="checkbox"/> Rexistro 7	Descrición Rexistro 7	1	1	1	1,00	1,00	100,00	<input checked="" type="checkbox"/>	Home	09/04/2020 14:59:43	abril	Abril	12:59:43	14:59:43	13:10:06

mail_wizard_invite_res_partner_rel	mail_wizard_invite_res_partner_rel	message_attachment_rel	message_attachment_rel	messages	no_notifications
odoo_basico_informacion	odoo_basico_informacion	odoo_basico_informacion	odoo_basico_informacion	odoo_basico_informacion	odoo_basico_informacion
Columns	Columns	Columns	Columns	Columns	Columns
Constraints	Constraints	Constraints	Constraints	Constraints	Constraints
Indices	Indices	Indices	Indices	Indices	Indices
Rules	Rules	Rules	Rules	Rules	Rules
Triggers	Triggers	Triggers	Triggers	Triggers	Triggers
odoo_basico_lineapedido	odoo_basico_lineapedido	odoo_basico_lineapedido	odoo_basico_lineapedido	odoo_basico_lineapedido	odoo_basico_lineapedido
odoo_basico_lineapedido_informacion	odoo_basico_lineapedido_informacion	odoo_basico_lineapedido_informacion	odoo_basico_lineapedido_informacion	odoo_basico_lineapedido_informacion	odoo_basico_lineapedido_informacion

No meu caso teño configurado o usuario co "timezone" Europe/Madrid:

- No horario de inverno a hora en Europe/Madrid e UTC+1
- No horario de verán a hora en Europe/Madrid e UTC+2

Administrator - Odoo x +


localhost:8069/web#id=2&action=68&model=res.users&view_type=fo

Ajustes Tablero Usuarios y compañías Traducciones Opciones Generales

Tablero / Usuarios / Administrator

Editar Crear

Enviar instrucciones de restablecimiento de la contraseña



Administrator
odoo@odoo.com

Permisos de acceso Preferencias

Localización

Idioma Spanish / Español
Zona horaria Europe/Madrid

Mensajería y redes sociales

Gestión de Manejar por correos electrónicos

Conversión de timezone UTC a timezone do usuario

Debido ao explicado no punto anterior, cando accedemos mediante código a un campo datetime, o valor que leemos está en horario UTC.

Se queremos traballar coa hora en formato texto, temos que facer a conversión a zona horaria (timezone) do usuario.

Para facer a conversión do timezone UTC ao timezone do usuario imos utilizar **pytz** por tanto o primeiro que temos que facer é engadir a librería:

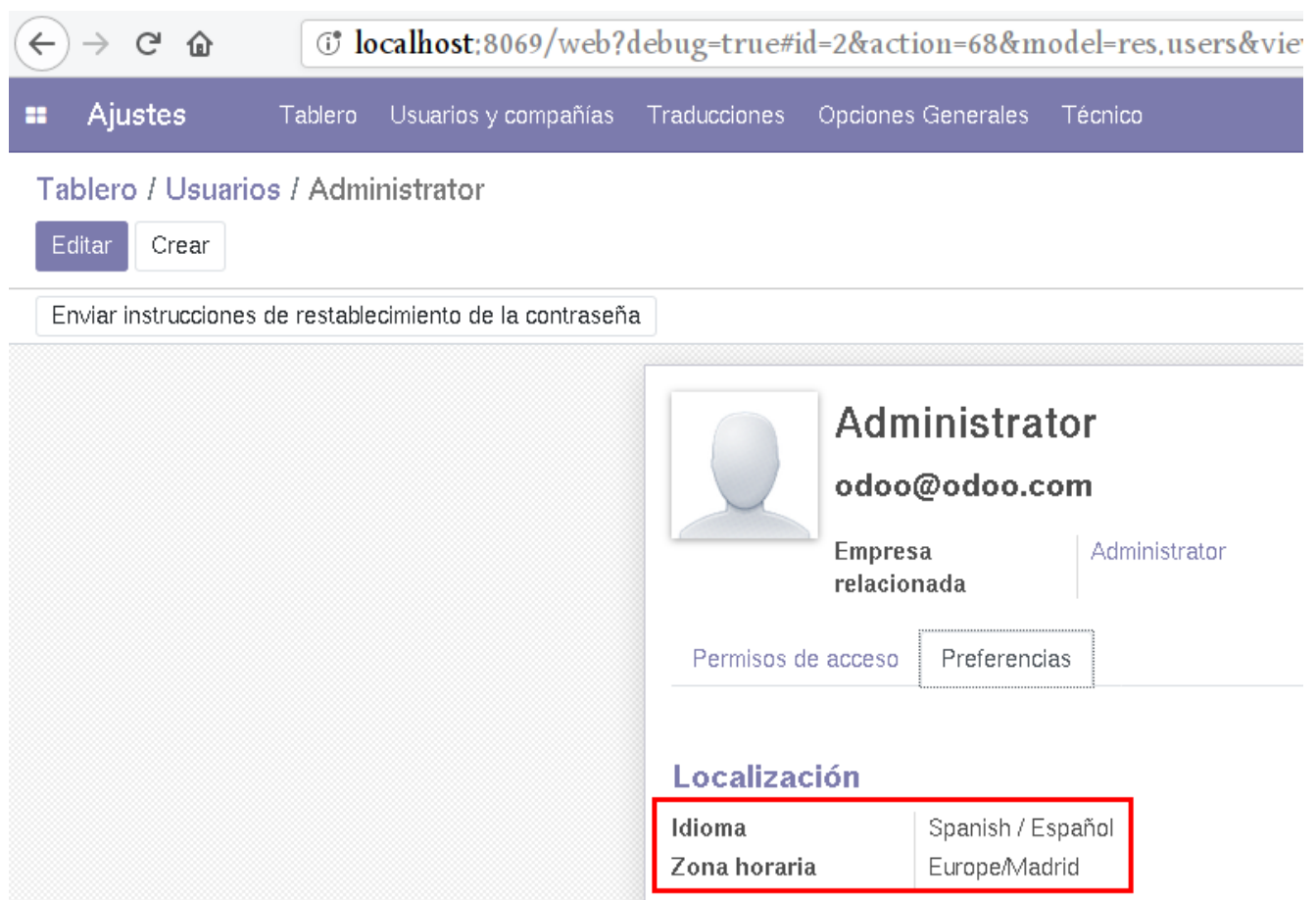
```
import pytz
```

O método **def** `convirte_data_hora_de_utc_a_timezone_do_usuario(self, data_hora_utc_object):`

Recibe como parámetro o obxeto que ten o campo **data_hora** (coa hora no timezone UTC) que queremos converter ao timezone do usuario.

```
def convirte_data_hora_de_utc_a_timezone_do_usuario(self, data_hora_utc_object):
    usuario_timezone = pytz.timezone(self.env.user.tz or 'UTC') # obter a zona
    return pytz.UTC.localize(data_hora_utc_object).astimezone(usuario_timezone)
# para usar pytz temos que facer import pytz
```

Almacenamos en **usuario_timezone** o timezone do usuario que obtemos mediante **self.env.user.tz** da configuración do usuario:



The screenshot shows the Odoo web interface for user configuration. The browser address bar indicates the URL: `localhost:8069/web?debug=true#id=2&action=68&model=res.users&vie`. The navigation menu includes 'Ajustes', 'Tablero', 'Usuarios y compañías', 'Traducciones', 'Opciones Generales', and 'Técnico'. The breadcrumb trail is 'Tablero / Usuarios / Administrator'. There are 'Editar' and 'Crear' buttons. A message box says 'Enviar instrucciones de restablecimiento de la contraseña'. The user profile section shows a placeholder for a profile picture, the name 'Administrator', and the email 'odoo@odoo.com'. Below this, there are tabs for 'Permisos de acceso' and 'Preferencias'. The 'Localización' section is highlighted with a red box and contains the following settings:

Idioma	Spanish / Español
Zona horaria	Europe/Madrid

Se nesta pantalla non ten asignada timezone poñemos por defecto UTC (`self.env.user.tz` or `'UTC'`). Poderíamos poñer outro por exemplo Europe/Madrid

Finalmente facemos a conversión do timezone UTC ao timezone do usuario mediante:

```
pytz.UTC.localize(data_hora_utc_object).astimezone(usuario_timezone)
```

Acceso dende código aos campos datetime

Para facer probas coas horas, temos definido o campo `data_hora` que é un datetime e outros 3 campos tipo Char onde almacenaremos horas en formato texto.

```
data_hora = fields.Datetime(string="Data e Hora", default=lambda self: fields.Date
hora_utc = fields.Char(compute="_hora_utc",string="Hora UTC", size=15, store=True
hora_timezone_usuario = fields.Char(compute="_hora_timezone_usuario",string="Hora
hora_actual = fields.Char(compute="_hora_actual",string="Hora Actual", size=15, s
```

- En **hora_utc** almacenaremos a hora tal cual a leemos do campo **data_hora** (é dicir quedará en horario UTC), como podemos ver no marco azul da imaxe anterior.

```
@api.depends('data_hora')
def _hora_utc(self):
    for rexistro in self: # A hora se almacena na BD en horario UTC (2 horas me
        rexistro.hora_utc = rexistro.data_hora.strftime("%H:%M:%S")
```

- En **hora_timezone_usuario** almacenaremos a hora do campo **data_hora** convertida a zona horario do noso usuario, como podemos ver no marco verde da imaxe anterior. Despois veremos como se converte.

```
@api.depends('data_hora')
def _hora_timezone_usuario(self):
    for rexistro in self:
        rexistro.actualiza_hora_timezone_usuario_dende_boton_e_apidepends()
```

En **hora_actual** almacenaremos a hora do momento na que estamos usando a

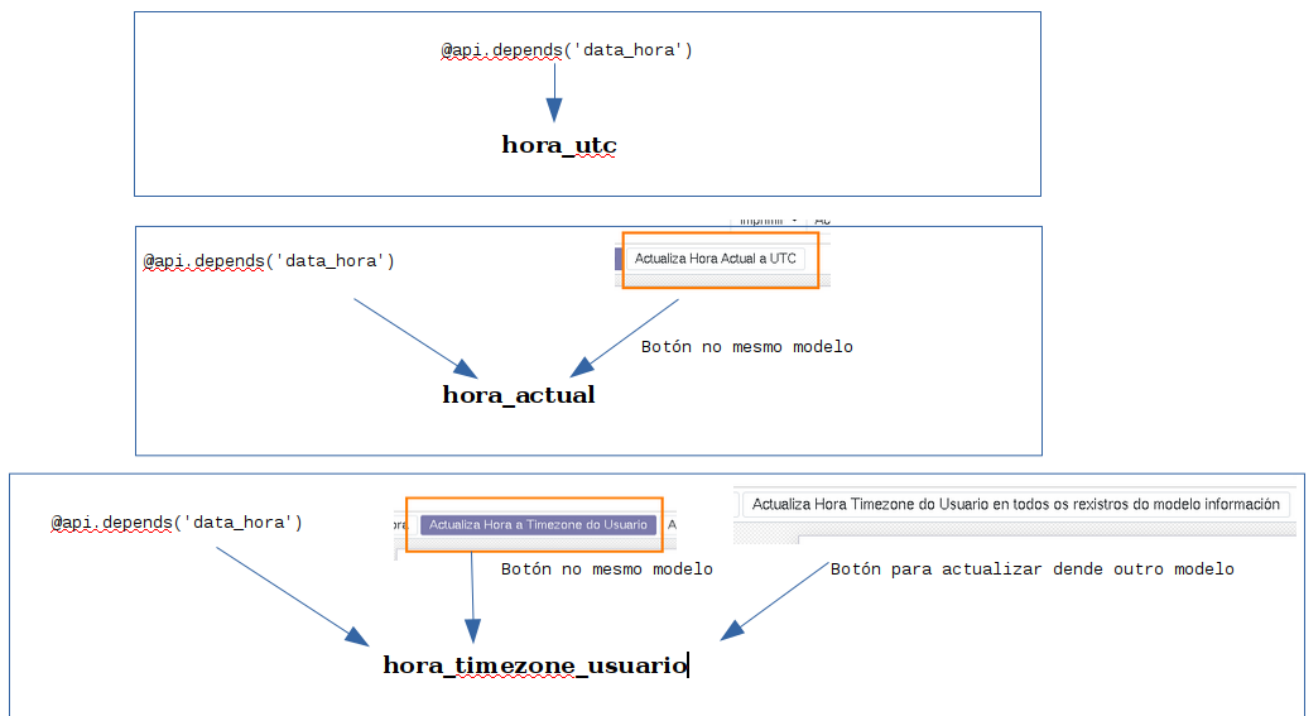
- En **nora_actual** almacenaremos a nora do momento no que estamos usando a función **fields.Datetime.now()**. Vemos que queda **gravada en horario UTC**, se quixesemos teriamos que convertila.

```
def actualiza_hora_actual_UTC(self): # Esta función é chamada dende un boton de
    for rexistro in self:
        rexistro.hora_actual = fields.Datetime.now().strftime("%H:%M:%S")
    # Grava a hora en UTC, se quixesemos poderiamos usar a función _convirte_d

@api.depends('data_hora')
def _hora_actual(self):
    for rexistro in self:
        rexistro.actualiza_hora_actual_UTC()
```

A continuación explico máis polo miúdo cando e como se actualizan os campos **hora_utc**, **hora_actual** e **hora_timezone_usuario**, xa que o código é chamado dende diferentes puntos do aplicativo.

A modo de resumo podes ver a seguinte imaxe:



Actualizar o campo **hora_utc**

O campo **hora_utc** só se actualiza cando cambiamos o valor do campo **data_hora**. Isto

conseguimos mediante:

```
@api.depends('data_hora')
def _hora_utc(self):
    for rexistro in self: # A hora se almacena na BD en horario UTC (2 horas menos no
        rexistro.hora_utc = rexistro.data_hora.strftime("%H:%M:%S")
```

Actualizar o campo `hora_actual`

```
def actualiza_hora_actual.UTC(self): # Esta función é chamada dende un boton de informac
    for rexistro in self:
        rexistro.hora_actual = fields.Datetime.now().strftime("%H:%M:%S")
    # Grava a hora en UTC, se quixesemos poderíamos usar a función _convirte_data_ho
```

O campo `hora_actual` actualízase mediante a chamada ao método `actualiza_hora_actual.UTC`.

Este método o chamaremos dende dous puntos distintos do aplicativo como vemos continuación:

- **Primeiro punto:** Ao igual que o campo `hora_utc` o campo `hora_actual` actualízase cando cambiamos o valor do campo `data_hora`. Mediante:

```
@api.depends('data_hora')
```

Onde no método `_hora_actual` chamamos ao método `actualiza_hora_actual.UTC`

```
@api.depends('data_hora')
def _hora_actual(self):
    for rexistro in self:
        rexistro.actualiza_hora_actual.UTC()
```

- **Segundo punto:** Cando clickamos no botón da vista de información.

Para elo definimos un botón no header do arquivo **informacion.xml** que chama ao método **actualiza_hora_actual.UTC**

```
<button name="actualiza_hora_actual.UTC" type="object" string="Actualiza Hora Actual a l
```

Menú Nivel 0 Menú Nivel 1

A acción de Información / Rexistro 1

Editar Crear Imprimir Acción

Ver Contexto Enviar Email Localización da Data hora Actualiza Hora a Timezone do Usuario Actualiza Hora Actual a UTC

Datos persoais	Datos Numéricos
Titulo	Rexistro 1
A Descrición	Descrición Rexistro 1
¿Autorizado?	<input checked="" type="checkbox"/>
Sexo	Home
Data	10/10/2020
Mes castelán	abril
Mes Galego	Abril
Data e Hora	08/04/2020 14:59:43
Hora UTC	12:59:43
Hora Timezone do Usuario	14:59:43
Hora Actual	10:53:34
Arquivo Adxunto	
Foto	
Foto	

O método **_hora_actual** non o podemos chamar dende a vista, xa que o usamos co **@api.depends**.

Por iso usamos o método **actualiza_hora_actual.UTC** (público) que podemos chamal dende os dous puntos vistos anteriormente dende **informacion.py** e dende a vista.

Actualizar o campo **hora_timezone_usuario**

```
# Esta función será chamada dende a función actualiza_hora_timezone_usuario_dende_boi
# dende pedido.py (Cando insertamos os valores do template self.env.user.tz non ten
# o botón en pedido.py é para actualizar todos os rexistros masivamente dende outro
def actualiza_hora_timezone_usuario(self, obxeto_rexistro):
    obxeto_rexistro.hora_timezone_usuario = self.convirte_data_hora_de_utc_a_timezone
```

O campo `hora_timezone_usuario` actualízase mediante a chamada ao método `actualiza_hora_timezone_usuario` onde vemos que chama ao método de conversión visto anteriormente `convirte_data_hora_de_utc_a_timezone_do_usuario`.

Temos que fixarnos que o método `actualiza_hora_timezone_usuario` ten dous parámetros:

- un é `self` (que como vimos no apartado "Acceso dende un modelo ao código doutr modelo", python o pasa automaticamente <https://pythontips.com/2013/08/07/the-self-variable-in-python-explained/>)
- o outro é `obxeto_registro`, que será o registro que queremos actualizar. Necesitamos pasarlle este segundo parámetro xa que un dos puntos dende onde queremos actualizar o campo `hora_timezone_usuario` será dende outro modelo (en concreto dende `pedido.py`)

Este método `actualiza_hora_timezone_usuario` chamáremolo dende tres puntos distintos do aplicativo como vemos a continuación:

- **Primeiro punto:** Ao igual que o campo `hora_utc` e o campo `hora_actual` o campo `hora_timezone_usuario` actualízase cando cambiamos o valor do campo `data_hora`. Mediante:

```
@api.depends('data_hora')
```

Onde no método `_hora_timezone_usuario` chamamos ao método `actualiza_hora_timezone_usuario_dende_boton_e_api` dende o que finalmente chamamos a `actualiza_hora_timezone_usuario` pasandolle como parámetro `self` xa que como a definición de ese método ten 2 parámetros (o primeiro xa llo pasa python e non pasamoslle o outro). Neste caso os dous fan referencia ao mesmo obxecto, xa que estamos no modelo "informacion.py", pero temos que pasarllo xa que a definición do método espera dous parámetros.

```
def actualiza_hora_timezone_usuario_dende_boton_e_api(self): # Esta función é chamada desde el boton e la api
    self.actualiza_hora_timezone_usuario(self) # lleva self como parametro por que a funcion actualiza_hora_timezone_usuario
    # porque usamos también actualiza_hora_timezone_usuario dende outro modelo (pedido.py)

@api.depends('data_hora')
def _hora_timezone_usuario(self):
    for registro in self:
        . . . . .
```



```
registro.actualiza_hora_timezone_usuario_dende_boton_e_apidepends()
```

- **Segundo punto:** Cando clickamos no botón da vista de información.

Para elo definimos un botón no header do arquivo **informacion.xml** que chama a método **actualiza_hora_timezone_usuario_dende_boton_e_apidepends**

```
<button name="actualiza_hora_timezone_usuario_dende_boton_e_apidepends" type="object" si
```

Menú Nivel 0 Menú Nivel 1

A acción de Información / Rexistro 1

Editar Crear Imprimir Acción

Ver Contexto Enviar Email Localización da Data hora Actualiza Hora a Timezone do Usuario Actualiza Hora Actual a UTC

Datos persoais	Datos Numéricos
Titulo	Rexistro 1
A Descrición	Descrición Rexistro 1
¿Autorizado?	<input checked="" type="checkbox"/>
Sexo	Home
Data	10/10/2020
Mes castelán	abril
Mes Galego	Abril
Data e Hora	08/04/2020 14:59:43
Hora UTC	12:59:43
Hora Timezone do Usuario	14:59:43
Hora Actual	10:53:34
Arquivo Adxunto	
Foto	
Foto	

- **Terceiro punto:** Dende un botón definido no header do arquivo **pedido.xml** que é a vista do modelo **pedido.py** que chama ao método **actualizadorHoraTimezone**

```
<button name="actualizadorHoraTimezone" type="object" string="Actualiza Hora Timezone d
```

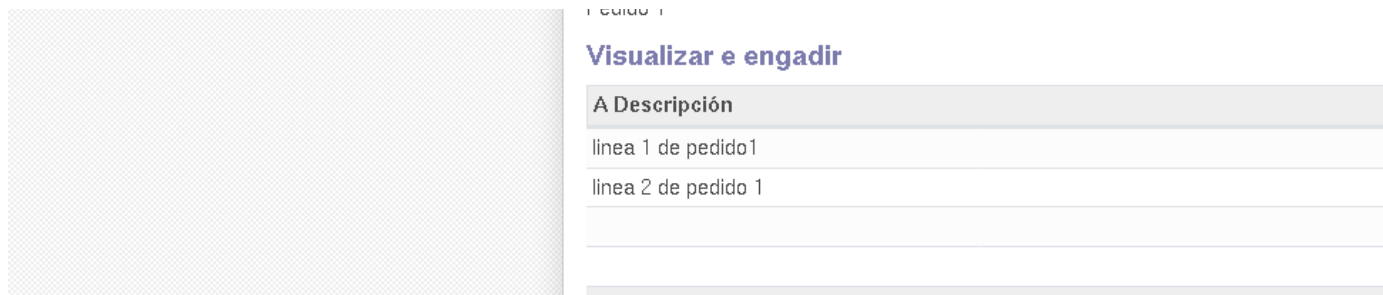
Menú Nivel 0 Menú Nivel 1

A acción de pedido / Pedido 1

Editar Crear Acción

Actualiza o campo Sexo do modelo información	Actualiza Hora Timezone do Usuario en todos os rexistros do modelo información
--	--

Pedido 1



A idea de usar este botón é dobre:

- Para ver como se executa código definido nun modelo dende outro modelo. (Onde vemos como temos que pasar o parámetro)
- Mediante este botón imos actualizar masivamente todos os rexistros de información para solucionar a seguinte problemática. Cando instalamos o módulo, os datos de proba importanse mediante XML(templates.xml). Nese momento o contexto de traballo (self.env.user), que utilizamos para facer a conversión (como vimos no punto "Conversión de timezone UTC a timezone do usuario"), non ten cargado o timezone do usuario, polo iso o campo **hora_timezone_usuario** queda coa hora en UTC. Como podemos comprobar na seguinte imaxe:

Menú Nivel 0 Menú Nivel 1															
A acción de Información															
<input type="button" value="Crear"/> <input type="button" value="Importar"/>															
<input type="text" value="Buscar..."/> <input type="button" value="Filtros"/> <input type="button" value="Agrupar por"/> <input type="button" value="★ Favoritos"/>															
1-6 / 6															
<input type="checkbox"/>	Título	A Descrición	Alto en centímetros	Longo en centímetros	Ancho en centímetros	Volume	Peso en Kg.s	Densidade	¿Autorizado?	Sexo	Data e Hora	Mes castelán	Mes Galego	Hora UTC	Hora Timezone do Usuario
<input type="checkbox"/>	Rexistro 6	Descrición Rexistro 6	4	4	4	64,00	4,00	6,25	<input type="checkbox"/>	Muller	28/04/2020 12:35:48	abril	Abril	10:35:48	10:35:48
<input type="checkbox"/>	Rexistro 5	Descrición Rexistro 5	4	4	4	64,00	4,00	6,25	<input type="checkbox"/>	Muller	28/04/2020 12:35:48	abril	Abril	10:35:48	10:35:48
<input type="checkbox"/>	Rexistro 4	Descrición Rexistro 4	4	4	4	64,00	4,00	6,25	<input checked="" type="checkbox"/>	Muller	28/04/2020 12:35:48	abril	Abril	10:35:48	10:35:48
<input type="checkbox"/>	Rexistro 3	Descrición Rexistro 3	3	3	3	27,00	3,00	11,11	<input checked="" type="checkbox"/>	Home	28/04/2020 12:35:48	abril	Abril	10:35:48	10:35:48
<input type="checkbox"/>	Rexistro 2	Descrición Rexistro 2	2	2	2	8,00	2,00	25,00	<input checked="" type="checkbox"/>	Home	28/04/2020 12:35:48	abril	Abril	10:35:48	10:35:48
<input type="checkbox"/>	Rexistro 1	Descrición Rexistro 1	1	1	1	1,00	1,00	100,00	<input checked="" type="checkbox"/>	Home	28/04/2020 12:35:48	abril	Abril	10:35:48	10:35:48

Como vimos o botón definido en **pedido.xml** chama ao método **actualizadorHoraTimezon** definido no modelo **pedido.py**

```
def actualizadorHoraTimezone(self):
    informacion_ids = self.env['odoo_basico.informacion'].search([])
    for rexistro in informacion_ids:
        self.env['odoo_basico.informacion'].actualiza_hora_timezone_usuario(rexistro)
```

Vemos como mediante:

```
informacion_ids = self.env['odoo_basico.informacion'].search([])
```

almacenamos en **informacion_ids** todos os rexistros do modelo **informacion**.

E mediante o bucle imos executando para cada rexistro o método **actualiza_hora_timezone_usuario** definido no modelo **informacion**.

Pasandolle como parámetro o propio rexistro. Neste punto non podemos usar **self** xa que fa referencia ao obxecto do modelo **pedido**.

Esta é a razón pola que definimos no modelo **informacion** un método que recibe com parámetro o obxecto a actualizar.

Se todas as chamadas a ese método foran feitas dende **informacion** poderíamos non definir parámetro xa que mediante **self** sabríamos que obxecto actualizar.

En resumo, usamos tres métodos distintos porque:

O método **_hora_timezone_usuario** non o podemos chamar dende a vista, xa que o usamo con **@api.depends**

Por iso usamos o método **actualiza_hora_timezone_usuario_dende_boton_e_apidepend** (público) que podemos chamalo dende os dous puntos vistos anteriormente dende **informacion.py** e dende a vista.

E utilizamos o método **actualiza_hora_timezone_usuario** porque necesitamos pasarlle com parámetro o rexistro a actualizar xa que o chamamos dende outro modelo (**pedido.py**). Por is na súa definición ten dous parámetros (**self, obxecto_rexistro**)