Rapport de projet Licence ADSILLH

Khal, par le groupe Khalzone

Yorick Barbanneau, Simon Crespeau, Florian Lassenay, Axel Danguin, Maxime Oçafrain

mars 2020

université de BORDEAUX

Table des matières

Introduction

Choix du projet

Khal a été préféré en raison du langage accessible(python), de sa taille raisonnable, de l'accessibilité des mainteneurs, de sa philosophie (licence libre MIT, non développement pour windows) et de notre interet pour son fonctionnement sobre.

Khal est un programme de **gestion d'agendas** en ligne de commande et TUI basé sur les standards de la RFC 5545 (Icalendars).

Fonctionnalités

Khal est capable de:

- Lire et écrire des évènements et des Icalendars dans un dossier.
- Ajouter, modifier et supprimer simplement des évènements, en ligne de commande et TUI.

Khal dépend de la version 3.4 ou supérieure de `python`. Il est accessible sur la majorité des systèmes d'exploitation à l'exception de Microsoft Windows.

Les évènements sont enregistrés dans une base de données **SQLite** afin d'optimiser les traitements.

Quelques chiffres

- https://lostpackets.de/khal
- https://github.com/pimutils/khal
- Création du projet en **août 2014**
- Un peu plus de **2000 commits**
- **59 contributeurs**

Khal n'est donc pas un très gros projet qui est toutefois développé et maintenu avec sérieux, rassemblant une communauté restreinte mais active et fidèle. **Khal** se voit de surcroît être implémenté dans divers modules pour d'autres logiciels tels que des gestionnaires de bureau ou barres de tâches.

Communauté

Khal fait partie de pimutils, une organisation regroupant quatre programmeurs et mainteneurs de projets voués à la gestion d'information personnelle telle que les contacts, les agendas, les todolists…

geier est le mainteneur principal du projet. Neurophysicien (donc pas développeur professionnel), il travaille sur le logiciel pendant son temps libre.

Ecosystème pimutils

Les programmes faisant partie de **pimutils** n'ont pas les mêmes buts, mais ont été pensé pour être chacun la pièce d'un écosystème laissant la maîtrise à l'utilisateur sur ses informations.

khal dans pimutils

Khal

Khal, utilisable de manière autonome dans un environnement local, est aussi capable de se synchroniser avec des serveurs CalDAV au travers de **Vdirsyncer**.

Vdirsyncer

Vdirsyncer stocke les informations synchronisées dans des dossiers locaux, au format `.ics` pour les calendiers.

Tests automatisés

Khal est "livré" avec un ensemble de tests, destinés à vérifier le bon fonctionnement des différents objets et fonctions du logiciel. Les tests unitaires sont réaliés avec `pytest` tandis que les tests de compatibilité sont réalisés avec `tox`.

Organisation du travail

Dépôt git

Une organisation Khalzone a été créée sur Github : Khalzone avec des droits mainteneurs équivalents pour chacun. Nous y avons créé un fork du projet **Khal**. Nos études ont été réalisées sur des branches de cette fourchette et c'est le master qui est proposé en pull request au projet principal.

Lorque nous voulons faire des demandes d'intégration (ou *merge requests*), nous le faisons via **Khalzone**.

Outils utilisés

(Voir annexe A) Nous avons exploité certaines possibilités de got telles que git bisect
Pour le debug nous avons fait appel à pdb, ipdb et pudb

Les tests

Khal est "livré" avec un ensemble de tests, destinés à vérifier le bon fonctionnement des différents objets et fonctions du logiciel. Les tests unitaires sont réaliés avec `pytest` tandis que les tests de compatibilité sont réalisés avec `tox`.

Voir en annexe B un exemple

Contenu

Non

Test :

```
@app.route('/management')
def redirect_management():
    res, client, chambre, conso, bar = liste_client()
    nres, nclient, nchambre, nconso, nbar = select_col_name()
    if isinstance(res, list):
        return render_template("management-board.html", rows_res
        ↪  = res, rows_client = client, rows_chambre = chambre,
        ↪  rows_conso = conso, rows_bar = bar, nres= nres,
        ↪  nclient = nclient, nchambre= nchambre, nconso
        ↪  =nconso, nbar= nbar )
    else: return render_template("management-board.html")

@app.route('/menu_bar')
def redirect_menu_bar(session=session):
    bar = liste_bar()
    return render_template("menu_bar.html", session=session,
     ↪  bar=bar)
```

Conclusion

Oui

Annexes

Annexe A

git bisect

`git bisect` est une fonctionnalité de `git` qui permet de découvrir quel commit a introduit un bug (basée sur une méthode de dichotomie)

pdb

pdb permet de debugger plus facilement son programme Python, il permet de faire facilement de l'introspection, du pas à pas etc.

Les commandes

Une fois le point d'arrêt atteint, il est possible de réaliser plusieurs actions via des commandes.

Commande générales

- `help`(h) : afficher l'aide

Pile d'exécution

- `where` (w) : affiche la pile d'exécution
- `list` (l) : affiche les lignes entourant l'instruction en cours. Il est possible d'afficher une liste plus longue avec `ll`. Il est aussi possible de spécifier les lignes du fichier courant à afficher avec la commande `list <ligne_debut>,<lignes_fin)`
- up (u) : remonte dans le contexte parent
- down (d) : descend dans le contexte enfant

Contrôle d'exécution

- `continue` (c) : continue l'execution normale du programme jusqu'au prochain point d'arrêt.
- `next` (n) : continue l'exécution jusqu'à la prochaine ligne de la fonction courante.
- `step` (s) : continue l'exécution jusqu'à prochaine instruction. Contrairement à `step`, `next` s'arrête après le retour de la fonction appelée.
- `return` (r) : continue l'exécution jusqu'à la prochaine exécution `return`

Une documentation plus complète est disponible ici en Anglais Modifier cette section

Pudb

Il existe aussi pudb qui tourne … avec urwid ø/

Taper *?* pour avoir les infos sur les commandes

```
import pudb ; pu.db # pu.db étant le breakpoint
```

Annexe B

Exemple de test

Voici comment se déroule un ajout d'évènement dans **khal** ainsi que l'affichage de ce dernier dans le terminal.

```
$ khal new -a adsillh 09/12/2019 14:00 18:00 Rendez-vous Projet

$ khal calendar
      lu ma me je ve sa di     lundi, 09/12/
déc.  25 26 27 28 29 30  1     14:00-18:00 Rendez-vous Projet
       2  3  4  5  6  7  8     18:00-19:30 Sport
       9 10 11 12 13 14 15
      16 17 18 19 20 21 22
      23 24 25 26 27 28 29
```

Annexe C

Issue 705

Changement des couleurs de ikhal à travers le fichier de config

https://github.com/pimutils/khal/issues/705

Les schémas de couleurs ne donnent pas satisfaction.

> Colors color theme should be changeable through the color theme. This should be easily doable by literal_evaling a string and adding it to the selected color theme (as urwid iterates over the palette list and later attributes overwrite earlier ones with the same name). Proper documentation might be an issue.

Traduction

Les couleurs du theme couleur devraient etre modifiables dans/au travers du thème de couleur. Ceci devrait etre facilement réalisable en plaçant une variable de type et en l'ajoutant dans le thème désiré. (comme urwid itere au-dessus de la liste de la palette et après attribue par-dessus les précédentes avec le même nom) Une documentation plus propre peut etre une solution

Documentation

Ajouter d'un asciiart pour aider à la compréhension. Des parametres de couleurs et leur impact dans ikhal.

- Dans le fichier de configuration pour les utilisateurs.
- Dans le source colors.py pour les contributeurs.

Paramètres intermédiaires

Création de trois parametres intermédiaires dans le fichier de configuration. (cela ne semble pas pertinent de demander aux utilisateurs s'interresser aux 29 paramètres actuels)

Création de deux nouveaux thèmes

Deux thèmes avec des paramètres triés et classés (certains semblent être des doublons, d'autres ne jamais être utilisés)

Thème user

Ce thème remplace les valeurs nommées des couleurs Seuls trois parametres demeurent modifiables dans le fichier de configuration.

Thème 256-colors

ce thème est plus lourdement modifié pour pouvoir transmettre des valeurs 256 couleurs à urwid. Les quatre variables sont associées à un code 256 couleurs. L'utilisation d'un code 256 couleurs plutot qu'un nom de couleur entraine un comportement différent d'urwid : les couleurs définies ne sont pas impactées par le choix de thème de couleur de console. Ce thème a pour avantage de pouvoir piloter finement les couleurs choisies et de pouvoir identifier les champs non colorés.

[mode_256couleurs]mode(img/256couleurs.jpeg)]]

Questions posées

- concernant les parametres apparemment non utilisés a été posée https://github.com/pimutils/khal/
  Recherche dans l'historique d'ou viennent ces parametres git blame khal/ui/colors.py

Notes de travail en cours

Mode 256 couleurs

- Pas mal de modifs sur trois fichiers. Encore des incertitudes sur le code.
- Trouver les éléments impactés par les 29 paramètres des thèmes de color.py
- **khal/settings/khal.spec** définit les themes disponibles dans **khal/ui/colors.py** et désigne celui utilisé par défaut.
- Un nouveau theme *colors256* à khal/ui/colors.py en utilisant le codage xterm compatible avec urwid , le moteur de rendu.

khal/ui/init.py contient les fonctions apellant urwid ainsi qu'un descriptif en ASCCI-art de l'interface.

Toutefois ce dernier ne permet pas d'identifier tous les composants colorés dans khal/ui/colors.py Un descriptif plus parlant rendrait le paramétrage plus aisé (ascii art évoqué dans le fil de discussion)

La fonction urwid.WidgetWrap.__init__() permet d'appeler les attibuts. En l'enrichissant avec la fonction urwid.AttrMap() on peut appeler la couleur souhaitée : exemple : `urwid.WidgetWrap.__init__(self, urwid.AttrMap(self.pile, 'popupbg'))`

khal/ui/widget.py

Dans les classes pour modifier les couleurs

class Choice(urwid.PopUpLauncher):

```
234
urwid.PopUpLauncher.__init__(self, self.button)
urwid.PopUpLauncher.__init__(self, urwid.AttrMap(self.button, 'alert'))
```

class ChoiceList(urwid.WidgetWrap):

```
urwid.WidgetWrap.__init__(self, urwid.AttrMap(fill, 'popupbg'))
urwid.WidgetWrap.__init__(self, urwid.AttrMap(fill, 'alert'))
```

Coquille dans la désignation de la variable de couleur "editfc" ou "edit focused" ? ne serait-il pas pertinent de n'en utiliser qu'une ? Question posée au mainteneur.

```
395
- self._original_widget = urwid.AttrMap(EditWidget(*args, **kwargs),
'edit', 'editf')
+ self._original_widget = urwid.AttrMap(EditWidget(*args, **kwargs),
'edit', 'edit focused')
```

Sans impact L'appel des méthodes dans les lignes suivantes ne semble pas avoir d'impact (modification du parametre de couleur ou introduction de set_attr_map()

khal/ui/widgets.py class ValidatedEdit(urwid.WidgetWrap):

```
413 self._original_widget.set_attr_map({None: 'alert'})  # undefined use
414 self._original_widget.set_focus_map({None: 'alert'}) # undefined use
```

class DurationWidget(urwid.WidgetWrap):

```
495 urwid.WidgetWrap.__init__(self, urwid.AttrMap(self.columns,
'theme color'))
```

class AlarmsEditor(urwid.WidgetWrap):

```
529 urwid.WidgetWrap.__init__(self, urwid.AttrMap(self.columns,
'theme color'))
```

class AlarmsEditor(urwid.WidgetWrap):

```
547 urwid.WidgetWrap.__init__(self, urwid.AttrMap(self.pile, 'theme
color'))
```

class FocusLineBoxWidth(urwid.WidgetDecoration, urwid.WidgetWrap):

```
608 urwid.WidgetDecoration.__init__(self, urwid.AttrMap(widget, 'alert-

for what ?-'))
609 urwid.WidgetWrap.__init__(self, urwid.AttrMap(self._all, 'for what

614 self._all.contents[0] = (self._topline_focus, ('pack-
for what ?-',
None)) 615 inner.contents[0] = (self._vline_focus, ('given-
for what ?-',
1, False)) 616 inner.contents[2] = (self._vline_focus, ('given-
for what
?-', 1, False)) 617 self._all.contents[2] = (self._bottomline_focus,
('pack- for what ?-', None))
```

class FocusLineBoxColor(urwid.WidgetDecoration, urwid.WidgetWrap):

```
- urwid.WidgetWrap.__init__(self, self._all)
- urwid.WidgetDecoration.__init__(self, widget)
+ urwid.WidgetWrap.__init__(self, urwid.AttrMap(self._all, 'alert'))
+ urwid.WidgetDecoration.__init__(self, urwid.AttrMap(widget, 'alert'))
```

```
659 self._middle.contents[0][0].set_attr_map({None: 'frame focus color'
660 self._all.contents[0][0].set_attr_map({None: 'frame focus color'})
661 self._all.contents[2][0].set_attr_map({None: 'frame focus color'})

663 self._middle.contents[0][0].set_attr_map({None: 'frame'})
664 self._all.contents[0][0].set_attr_map({None: 'frame'})
665 self._all.contents[2][0].set_attr_map({None: 'frame'})
```

class FocusLineBoxTop(urwid.WidgetDecoration, urwid.WidgetWrap):

```
673
- urwid.WidgetWrap.__init__(self, self._all)
+ urwid.WidgetWrap.__init__(self, urwid.AttrMap(self._all, 'alert'))
673
- urwid.WidgetDecoration.__init__(self, widget)
+ urwid.WidgetDecoration.__init__(self, urwid.AttrMap(widget, 'alert'))
```