

Logiciels

Sélection de logiciels que j'utilise

- [Mesure de soi](#)
 - [La marche : Nextcloud avec GpxPod, Workflow et Analytics](#)
 - [Le poids : OpenScale](#)
 - [Le sommeil : SleepyHead et OSCAR](#)
 - [Montre connectée : Gadgetbridge](#)
 - [Une pointeuse avec Ktimetracker](#)
- [Gestion de documents](#)
 - [Espace de stockage : Nextcloud](#)
 - [Gestion électronique de documents : Mayan EDMS](#)
 - [Inventaire de livres physiques : Inventaire.io](#)
 - [Wiki : Bookstack et DokuWiki](#)
 - [Gérer sa bibliothèque électronique : Calibre](#)
 - [Signature électronique de documents : Documenso](#)
- [Vie de famille](#)
 - [Pour gérer les menus et les courses : Grocy](#)
 - [Pour une wishlist : Wishthis](#)
 - [Pour un père Noël secret : SecretSanta](#)
 - [Généalogie : Gramps web](#)
- [Organisation](#)
 - [Todo list](#)
- [Divers](#)
 - [Firefox](#)
 - [Pour les screencasts : Vokoscreen-ng](#)
 - [Pour surveiller des pages web : urlwatch](#)

- [Mon shell : zsh](#)

Mesure de soi

La marche : Nextcloud avec GpxPod, Workflow et Analytics

Pour ma santé, je dois faire du sport. J'ai choisi la marche rapide parce que ça peut se faire à peu près n'importe où, n'importe quand, que le matériel n'est pas hors de prix et que contrairement au *footing*, ça ne me flingue pas les articulations.

Garmin Forerunner 110

J'ai acheté d'occasion une montre Garmin Forerunner 110 pour suivre mes traces GPS et mes stats (vitesse, longueur du parcours, fréquence cardiaque, etc). Elle se branche à l'ordinateur avec un câble et se présente comme une clé USB. Très simple et très pratique.

Nextcloud et ses applications

[GpxPod](#), pour peu qu'on installe le logiciel [gpsbabel](#) sur le serveur, est capable de convertir les traces `.fit` fournies par la montre Garmin en `.gpx`, format standard pour les traces GPS et permettra ensuite de voir le parcours contenu dans ce fichier, avec une jolie carte venant d'[OpenStreetMap](#).

Je me sers d'un script [Workflow](#) perso pour extraire des données du fichier `.gpx` pour les mettre dans un fichier [CSV](#) qui sera lu par l'application [Analytics](#) pour me faire des graphes (fréquence cardiaque, vitesse moyenne, etc).

Garmin Forerunner 610

La Forerunner 110 a un bracelet en caoutchouc qui peut claquer et qui n'est malheureusement pas remplaçable (la montre fait partie intégrante du bracelet). Certes, j'en ai retrouvé une autre d'occasion, mais le bracelet a lui aussi cassée. J'ai donc cherché une autre montre qui fait le même boulot, mais avec un bracelet changeable, et qui soit encore assez ancienne pour ne pas nécessiter une application Android.

J'ai trouvé, toujours d'occasion, une Garmin Forerunner 610. Elle se connecte à l'ordinateur avec un petit *dongle* USB, mais pour récupérer les informations de la montre, il faut installer un logiciel, [antfs-cli](#), et sa dépendance, [openant](#).

Antfs-cli récupère les fichiers `.fit` de la montre dans un sous-dossier de `~/.config/antfs-cli`, il n'y a plus qu'à les envoyer à Nextcloud comme précédemment et c'est bon.

Mesure de soi

Le poids : OpenScale

Pour surveiller mon poids, j'utilise l'application Android [openScale](#) qui se connecte en bluetooth à ma balance Sanitas SBF70 (balance recherchée à partir des balances supportées par openScale).

Bonus : la balance calcule aussi d'autres trucs, comme le pourcentage de masse grasse et osseuse.

Bug connu : openScale n'arrive plus à se connecter à ma balance à cause du passage à Android 11 de mon téléphone. J'ai encore un vieux téléphone qui peut se connecter, donc je récupère mes données d'un côté, j'exporte, j'envoie ça sur mon téléphone de tous les jours avec [KDE Connect](#) et j'importe. C'est un peu fastidieux, mais je ne fais pas ça tous les jours non plus.

Le sommeil : SleepyHead et OSCAR

J'ai eu des problèmes de sommeil dus à des [hypopnées](#), ce qui m'a amené à utiliser un appareil à [pression positive continue](#) (PPC) pendant la nuit. Un machin qui envoie de l'air sous pression dans le nez et/ou la bouche.

Ces appareils produisent de la donnée lors de leur utilisation : pression envoyée, temps d'utilisation, réveils de l'utilisateur, etc. Ces données sont envoyées à l'entreprise qui vous appareille, mais vous ne pouvez généralement pas les consulter.

C'est là qu'intervient [SleepyHead](#), et maintenant son fork [OSCAR](#) (forké pour cause d'abandon de SleepyHead). Installez-le, donnez-lui les données de la carte SD de l'appareil et hop, vous aurez vos données avec des graphes en prime.

Mesure de soi

Montre connectée : Gadgetbridge

J'ai eu pendant un temps un *Xiaomi MiBand 2*, petite montre connectée qui, entre autres choses, compte le nombre de pas effectués tous les jours et surveille le sommeil.

Cela fonctionne avec une application propriétaire, mais aussi avec [Gadgetbridge](#), qui permet d'avoir des graphes, de gérer les réveils, de transférer les notifications sur la montre...

Je ne me sers plus de cette montre connectée, mais je conseille toujours Gadgetbridge ☐☐

NB : Gadgetbridge gère plusieurs autres modèles de montres connectées.

Une pointeuse avec Ktimetracker

Pour savoir combien de temps par jour je bosse, je me fais une pointeuse avec [Ktimetracker](#) : j'ai une tâche par jour (`1_lundi`, `2_mardi`, etc.), j'enclenche le chrono quand je commence ma journée et je le stoppe quand je fais une pause ou que j'ai fini ma journée.

Problème : ça nécessite d'ouvrir le logiciel pour voir où j'en suis (est-ce que je peux m'arrêter ?) et pour lancer/stopper le chrono.

J'ai enfin pris le temps de me pencher dessus et je me suis fait quelques scripts pour gérer tout ça plus simplement.

J'ai donc maintenant un [script](#) et un service *systemd*. Le script permet de lancer le chrono de la tâche du jour en cours, de stopper le chrono et de tourner en tâche de fond pour m'avertir que j'ai fait mes 8h de boulot et me proposer de stopper le chrono et de m'arrêter.

Le script communique avec Ktimetracker via [qdbus](#) et communique avec moi via `notify-send` (fourni par le paquet [libnotify-bin](#) : ça me fait des notifications comportant un bouton d'action sur lequel je peux cliquer pour lancer ou arrêter le chrono.

J'ai configuré des raccourcis claviers au niveau du système appelant les actions de lancement/stoppage du chrono, ainsi que celui qui affiche le chronomètre : j'ai toutes les informations/actions que je veux aisément disponibles et sans toucher ma souris ☐

Pour installer vous-même tout le bazar :

- installez les dépendances : `apt install ktimetracker qdbus libnotify-bin bc`
- installez le script quelque part sur votre système
- installez le [service systemd](#) dans `~/.config/systemd/user/ktstatus.service` (adaptez le chemin du script), puis faites `systemctl --user daemon-reload; systemctl --user enable --now ktstatus.service`
- créez-vous des raccourcis claviers vers `ktstatus.sh -g` pour lancer le chrono, `ktstatus.sh -s` pour l'arrêter et `ktstatus.sh` pour afficher le temps actuel.

Pour lancer Ktimetracker au démarrage sans qu'il n'apparaisse dans la barre des tâches ou dans les aperçus des bureaux, je le lance avec `kstart5 --iconify --skiptaskbar --skippager ktimetracker`. Faire `qdbus org.kde.ktimetracker /ktimetracker/MainWindow_1 org.qtproject.Qt.QWidget.hide` peu après (pas directement, y a un peu de latence) permet de le cacher. Il sera alors uniquement dans

la boîte des miniatures (ce qui me va très bien).

Gestion de documents

Espace de stockage : Nextcloud

Comme la plupart des gens, j'utilise [Nextcloud](#), avec le client pour pc pour synchroniser des dossiers et le client pour Android pour envoyer automatiquement des fichiers sur le serveur.

Gestion électronique de documents : Mayan EDMS

Bulletins de salaires, dossier scolaire du gamin, factures, modes d'emploi... Oui, tout ça peut se gérer en mettant ça dans un [Nextcloud](#) dans des dossiers et sous-dossiers, mais cela ne permet pas de lier des documents entre eux (genre la facture d'un appareil avec son mode d'emploi) ni de multi-classer un fichier (genre dans un dossier `Assurance` et en même temps dans un dossier `Appartement`).

C'est pour cela que j'ai installé [Mayan EDMS](#) sur mon serveur.

Attention : l'installation en elle-même ne pose pas particulièrement de problèmes mais il n'est pas simple à prendre en main. J'ai acheté le [bouquin](#) et ça m'a permis de mieux appréhender les différents aspects de Mayan. Ce n'était pas du luxe.

Dans les points intéressants du logiciel :

- possibilité de créer des index soit-même sur les critères qu'on veut
- possibilité de mettre des documents dans plusieurs « classeurs »
- possibilité de créer des méta-données sur les documents
- possibilité d'avoir plusieurs versions d'un même document
- recherche de doublons
- possibilité de créer des actions automatiques (genre classement automatique dans un classeur, renommage du document selon ses méta-données, etc)

Ma doc de mise à jour

Adapté de la [doc officielle](#) (lien archive.org car la doc d'upgrade n'est plus sur le [site de la doc officielle](#)).

```
export VERSION=4.9.1

```bash
supervisorctl stop mayan-edms-celery-beat \
 mayan-edms-gunicorn \
 mayan-edms-worker_a \
```

```

 mayan-edms-worker_b \
 mayan-edms-worker_c \
 mayan-edms-worker_d \
 mayan-edms-worker_e
cp /etc/supervisor/conf.d/mayan-edms.conf /etc/supervisor/conf.d/mayan-edms.conf.bak
sudo --user=mayan /opt/mayan-edms/bin/pip install --upgrade pip
sudo --user=mayan curl -s "https://gitlab.com/mayan-edms/mayan-edms/raw/v4.9/removals.txt" \
 --output /tmp/removals.txt &&
 sudo --user=mayan /opt/mayan-edms/bin/pip uninstall --requirement /tmp/removals.txt --yes
sudo --user=mayan /opt/mayan-edms/bin/pip install mayan-edms==${VERSION} &&
sudo --user=mayan MAYAN_MEDIA_ROOT=/opt/mayan-edms/media/ \
 /opt/mayan-edms/bin/mayan-edms.py platform_template supervisorctl \
 | sudo tee /etc/supervisor/conf.d/mayan-edms.conf &&
vimdiff -c 'map <F2> :diffget<cr>]czz | map <F3>]czz | syn off | windo set wrap | winc h' \
 /etc/supervisor/conf.d/mayan-edms.conf \
 /etc/supervisor/conf.d/mayan-edms.conf.bak

```

Petit bug pour la version 4.9.1 (j'ai sauté la 4.9, mais je pense que le souci aurait quand même été là) :

```

sudo --user=mayan wget https://gitlab.com/mayan-edms/mayan-edms/-
/raw/series/4.8/mayan/apps/mime_types/backends/python_magic.py -O /opt/mayan-
edms/lib/python3.11/site-packages/mayan/apps/mime_types/backends/python_magic.py
sudo --user=mayan /opt/mayan-edms/bin/pip install python-magic==0.4.27

```

```

sudo --user=mayan MAYAN_MEDIA_ROOT=/opt/mayan-edms/media/ \
 /opt/mayan-edms/bin/mayan-edms.py common_perform_upgrade &&
 supervisorctl start mayan-edms-celery-beat \
 mayan-edms-gunicorn \
 mayan-edms-worker_a \
 mayan-edms-worker_b \
 mayan-edms-worker_c \
 mayan-edms-worker_d \
 mayan-edms-worker_e

```

# Inventaire de livres physiques : Inventaire.io

J'utilise [Inventaire](#), directement sur le site officiel. C'est un des rares logiciels que je n'installe pas sur mon serveur, pour plusieurs raisons :

- la complexité (ce n'est pas juste un dossier php + une base de données, c'est un poil plus complexe)
- l'intelligence collective : à l'instar de Wikipédia et de Wikidata (qu'il utilise), les utilisateurices d'Inventaire vont enrichir et améliorer la base de données d'Inventaire et de Wikidata en corrigeant les notices de leurs livres ou en les créant.

Le plus : pouvoir se contenter de scanner le code-barre d'un livre pour l'ajouter à sa collection.

# Wiki : Bookstack et DokuWiki

## Bookstack

Vous l'avez sous les yeux, j'ai choisi [Bookstack](#) pour mon wiki.

Son organisation en étagères, livres, chapitres, pages me parle énormément, moi qui aime les livres (j'ai même un DUT métiers du livre). C'est une contrainte, mais cela permet aussi de se libérer de la gestion de la structure de son wiki.

En PHP/MySQL, il est simple à installer et extrêmement rapide.

Points intéressants :

- on peut créer des modèles de page
- on peut choisir d'écrire en [Markdown](#)
- on peut ajouter des méta-données qui pourront servir pour la recherche ([exemple](#))
- pense-bête : [la doc de mise à jour](#)

## Flux RSS des dernières pages

Documentation de base sur <https://www.bookstackapp.com/hacks/simple-page-rss-feed/>.

Créez un dossier dans `themes`, utilisez ce thème dans le fichier `.env` :

```
APP_THEME=le_nom_du_dossier
```

Créez un fichier `functions.php` :

```
<?php

use BookStack\Entities\Models\Page;
use Illuminate\Support\Facades\Route;

Route::get('/rss/pages/new', function() {
 $pages = Page::query()
```



```

->visible()
->orderBy('created_at', 'desc')
->take(20)
->get();

return response()->view('rss', ['pages' => $pages], 200, ['Content-Type' => 'text/xml']);
});

```

Créez un fichier `rss.blade.php` (le mien diffère de celui de la documentation), en changeant l'adresse et le titre :

```

<?xml version="1.0"?>
<rss version="2.0"
 xmlns:content="http://purl.org/rss/1.0/modules/content/"
 xmlns:wfw="http://wellformedweb.org/CommentAPI/"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:atom="http://www.w3.org/2005/Atom"
 xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
 xmlns:slash="http://purl.org/rss/1.0/modules/slash/"
 >
 <channel>
 <title>Derniers articles wiki Fiat Tux</title>
 <link>{{ url('/') }}</link>
 <atom:link href="https://wiki.fiat-tux.fr/rss/pages/new" rel="self"
type="application/rss+xml" />
 <description>Les derniers articles du wiki Fiat Tux</description>
 <lastBuildDate>{{ date(DATE_RSS) }}</lastBuildDate>
 @foreach($pages as $page)
 <item>
 <title>{{ $page->hasChapter() ? $page->chapter->name.' - ' : '' }}{{ $page->name
}}</title>
 <link>{{ $page->getUrl() }}</link>
 <description>{{ $page->getExcerpt() }}</description>
 <content:encoded><![CDATA[{{!! isset($page->renderedHTML) ?
preg_replace('/href="\s\/books/', 'href="https://wiki.fiat-tux.fr/books/', $page->renderedHTML)
: preg_replace('/href="\s\/books/', 'href="https://wiki.fiat-tux.fr/books/', $page->html)
!!}}]]></content:encoded>
 <pubDate>{{ $page->created_at->format(DATE_RSS) }}</pubDate>
 <guid>{{ $page->getUrl() }}#{{ $page->id }}</guid>
 </item>

```

```
@endforeach
</channel>
</rss>
```

Dans le menu Préférences > Personnalisation, ajoutez ceci dans « HTML personnalisé dans l'en-tête », en modifiant l'adresse et le titre :

```
<link rel="alternate" type="application/rss+xml" title="Derniers articles wiki Fiat Tux"
href="https://wiki.fiat-tux.fr/rss/pages/new" />
```

# Dokuwiki

Pour le boulot, je choisis plutôt [DokuWiki](#). Il a l'avantage d'être sans base de données, donc simple à sauvegarder et à restaurer.

Austère de base, on peut étendre ses fonctionnalités de façon très agréable avec ses nombreux plugins, comme lister automatiquement les pages d'une catégorie, faire des redirections, ajouter des méta-données qu'on pourra utiliser dans les listes automatiques...

# Gérer sa bibliothèque électronique : Calibre

## Calibre

[Calibre](#) est un logiciel multi-plateforme qui permet de gérer sa bibliothèque d'epub et autres livres électroniques, de les transférer vers sa liseuse et qui possède bien d'autres atouts (conversion de formats, récupération de flux RSS pour en faire des epub...).

Calibre permet de gérer plusieurs bibliothèques différentes (elles seront dans des dossiers séparés).

**NB :** si vous voulez accéder à votre bibliothèque calibre via un montage CIFS (Samba), [il faut ajouter l'option](#) `nobrl` [aux options de montage](#) sinon vous aurez des problèmes de `database is locked`.

## Calibre-web

[Calibre-web](#) permet d'exposer sa bibliothèque calibre au travers d'un site web.

Il permet le téléchargement, la consultation, le téléversement... en mode authentifié ou non, avec la possibilité d'affecter des droits différents selon les utilisateurs.

Très simple à utiliser, j'adore !

# Signature électronique de documents : Documenso

J'ai une petite activité d'auto-entrepreneur (quelque soit le nom qui est maintenant donné à ce genre d'activité, j'ai autre chose à faire que d'apprendre les nouvelles désignations par un gouvernement qui n'a que ça à faire que de trouver de nouveaux noms aux choses) et pour éviter à mes clients d'imprimer mon devis, le signer, le scanner et me l'envoyer (ou de s'amuser à mettre eux-mêmes un scan de leur signature sur le pdf), j'ai cherché une solution de signature électronique de documents.

Le grand gagnant est [Documenso](#).

Je l'avais déjà installé et testé, mais j'avais attendu avec impatience la compatibilité avec Nextcloud 29 de [Libresign](#), et cela semblait fonctionner (d'après mes tests). L'avantage aurait bien évidemment été l'intégration à Nextcloud, plus besoin de téléverser le fichier à signer, hop, pratique.

Mais un client n'a pas réussi à signer un document. Comme j'ai autre chose à faire que déboguer ça avec un client à l'autre bout du fil, j'ai décidé de rester sur Documenso.

## Installation

Bien évidemment, comme la plupart des logiciels de nos jours, la méthode d'installation suggérée est Docker. Pouah.

## Création d'un utilisateur dédié

```
adduser --disabled-login \
 --disabled-password \
 --home /var/www/documenso \
 --no-create-home \
 --shell /bin/false \
documenso
```

## Création d'une base de données PostgreSQL

```
sudo -u postgres createuser -P documenso
sudo -u postgres createdb -O documenso documenso
```

## Récupération du code

```
export VERSION=1.11.0
```

```
cd /var/www
git clone https://github.com/documenso/documenso.git
cd documenso
chown -R documenso: .
sudo -u documenso git checkout -b "v$VERSION" "v$VERSION"
```

## Configuration

```
sudo -u documenso cp .env.example .env
sudo -u documenso vimdiff -c 'syn off | windo set wrap | winc h' .env .env.example
```

NB : le `vimdiff` ne sert que lors d'une mise à jour, lors de l'installation initiale, contentez-vous d'éditer le fichier

## Compilation

```
sudo -u documenso npm ci
sudo -u documenso npm run build
```

## Migration de la base de données

```
sudo -u documenso npm run prisma:migrate-deploy
```

## Service systemd

```
cat <<EOF > /etc/systemd/system/documenso.service
[Unit]
Description=documenso
After=network.target
After=postgresql.service
```

```
Requires=postgresql.service
PartOf=postgresql.service

[Service]
Type=simple
User=documentso
Environment=PATH=/var/www/documentso/node_modules/.bin:/usr/local/bin:/usr/bin:/bin
WorkingDirectory=/var/www/documentso
ExecStart=/usr/bin/npm run start
TimeoutSec=15
Restart=always
SyslogIdentifier=documentso

[Install]
WantedBy=multi-user.target
EOF
systemctl daemon-reload
systemctl enable --now documentso
```

## Configuration Nginx

```
location / {
 include /etc/nginx/proxy_params;
 proxy_set_header Upgrade $http_upgrade;
 proxy_set_header Connection "upgrade";
 proxy_pass http://127.0.0.1:3001;
 proxy_redirect http://127.0.0.1:3001 https://documentso.example.org;
}
```


# Vie de famille

# Pour gérer les menus et les courses : Grocy

J'utilisais auparavant [Nexi](#) pour gérer ma liste de courses, mais avec plusieurs centaines d'articles existants, ça galérait pas mal, les ralentissement étaient notables. Je me suis donc mis en chasse d'un logiciel qui serait fait pour ça et je suis tombé sur [Grocy](#).

Grocy permet de gérer ses recettes, ses menus de la semaine, la liste de course, et même les stocks ou les corvées ! Truc appréciable, on peut désactiver les fonctionnalités non désirées dans la configuration et ainsi les faire disparaître de l'interface.

Parmi les plus de ce logiciel, je tiens à souligner :

- multi-utilisateurs : madame peut ajouter des trucs à la liste de course sans me demander
- possibilité de baser l'authentification sur LDAP : j'ai un annuaire LDAP à la maison, autant s'en servir
- possibilité d'imprimer les menus : parfait pour avoir les menus de la semaine sur le frigo 
- installation très simple : PHP/SQLite
- l'application Android est très pratique et évolue gentilement : elle vient par exemple d'ajouter la gestion des recettes. Plus que la gestion des menus dans l'appli et mon bonheur sera complet

Dans les moins :

- l'interface se recharge à chaque modification d'un truc, même si la modification est faite avec une requête AJAX
- j'aurais aimé la [possibilité de l'utilisation d'une base PostgreSQL](#), mais le développeur utilise des fonctions avancées de SQLite, ce qui fait qu'il devrait réécrire beaucoup de choses et je respecte sa volonté de ne pas se lancer là-dedans. Et ça fonctionne très bien comme ça (c'était juste que mes bases PostgreSQL sont sauvegardées au fil de l'eau grâce à [Barman](#), j'aurais bien voulu que ce soit le cas pour la base de données de mon Grocy).

Sources : <https://github.com/grocy/grocy>



# Pour une wishlist : Wishthis

Il est relativement difficile de me faire un cadeau. Beaucoup de gens savent que j'aime les bandes dessinées, mais comme j'ai une collection relativement monstrueuse, il est ardu de trouver la BD que je n'ai pas et qui me fera plaisir.

Pour simplifier cela, j'ai longtemps eu une page sur mon [blog](#) où je maintenais une *wishlist*. Ça a fait le job pendant longtemps, mais il y a toujours eu le risque de deux personnes qui choisissent le même cadeau pour la même occasion.

C'est là que [Wishthis](#) vient simplifier ça en permettant aux visiteurs de réserver un cadeau, ce qui le cache aux autres visiteurs, tout en laissant la personne qui a fait la *wishlist* dans l'ignorance de cette réservation. Nickel, quoi.

De plus, on peut fournir l'URL d'un cadeau sur un site marchand à Wishthis pour qu'il aille automatiquement chercher le nom de l'article et son image. J'imagine que ça ne fonctionne que si le site est bien codé, mais ça permet de se simplifier la vie pour la constitution de la *wishlist*.

Wishthis tourne en PHP/MySQL et l'installation est ultra simple mais on peut aussi se contenter d'utiliser l'[instance officielle](#) ☐

# Pour un père Noël secret : SecretSanta

J'ai cherché un projet qui fait tout ce qu'il faut, qui ne soit pas spécifique à une famille (y a des projets sur Github où la liste des gens est en dur dans un fichier !), et j'ai trouvé [SecretSanta](#).

On peut ajouter les gens à la main ou via un fichier CSV, et faire des exclusions. Les données sont chiffrées en base de données, ce qui est parfait pour que l'administrateur de l'instance n'aille pas regarder qui va lui offrir un cadeau ☐☐

Il n'y a pas de documentation d'installation. J'ai deviné comment l'installer (j'ai du un peu batailler) vu que c'est une application en PHP/Mysql<sup>1</sup>, mais je n'avais pas le temps d'écrire une documentation. Du coup, j'ai créé un [ticket](#).

Si vous ne voulez pas l'installer, vous pouvez utiliser l'[instance officielle](#).

<sup>1</sup> Je ne sais pas si on peut utiliser une autre base de données

Vie de famille

# Généalogie : Gramps web

Mes grands parents ont pris la peine d'établir leur arbre généalogique, ça serait dommage de laisser perdre ça.

C'est pourquoi je souhaite les recopier dans un logiciel libre de généalogie.

Mon choix s'est porté sur [Gramps](#), et plus particulièrement sur sa [version web](#).

J'ai fait un guide d'installation sur [cette page](#).

# Organisation

# Todo list

J'ai longtemps cherché un logiciel de todo list qui correspondrait parfaitement à mes besoins et je l'ai-enfin trouvé !

## Sleek

[Sleek](#) est une application multi-plateforme qui se base sur le format [todo.txt](#).

Elle coche toutes les cases de mes besoins :

- interface agréable (la manipulation via le terminal... disons que c'est pas le top pour ce genre de besoin, j'ai testé différents clients, c'était bof niveau praticité)
- gestion des priorités : LE critère essentiel pour moi
- possibilité de le réduire dans la boîte à miniatures du bureau pour toujours l'avoir sous la main sans encombrer la barre des tâches
- synchronisable : comme ça se base sur un fichier texte, rien de plus simple que de le stocker dans un répertoire synchronisé via [Nextcloud](#)

## Simpletask nextcloud

Pour accéder à ma todo list depuis mon téléphone, [Simpletask nextcloud](#) est parfait : il permet de se connecter facilement à mon serveur nextcloud et hop, je peux agir sur le même fichier que Sleek ☐☐

Notez qu'il existe deux autres versions de l'application :

- une version déconnectée
- une version se connectant à un serveur WebDAV générique (pas spécifiquement Nextcloud, quoi)

# Divers

# Firefox

Mon navigateur préféré... depuis sa version 1.0 !

Pour le télécharger : <https://firefox.com>.


## Mes extensions

- Anchors Reveal : affiche les ancres dans la page, idéal pour partager un lien qui vous amène directement au bon endroit
- auto-reader-view : active automatiquement le mode lecture sur les sites qu'on veut
- Awesome RSS : affiche l'icône RSS dans la barre d'URL quand un flux est disponible (Firefox, pourquoi l'as-tu enlevée ?)
- Bitwarden : gestionnaire de mot de passe (voir la [page de mon wiki](#) pour installer la réécriture du backend en Rust)
- Bookmarks Organizer : pour faire le ménage dans les marques-pages
- Bypass Paywalls Clean : fait sauter le *paywall* de certains sites
- Content-O-Matic : pour dire « zut » automatiquement aux bannières RGPD (ça ne fonctionne pas avec toutes)
- Dark Reader : indispensable, pour avoir un mode sombre même sur les sites ne le supportant pas
- Expire History By Days : vide automatiquement l'historique au-delà de la durée de rétention configurée
- Firefox Multi-Account Containers : pour avoir plusieurs *containers*, ce qui permet d'avoir plusieurs sessions différentes sur le même site mais dans des onglets (et *containers*) différents
- Firefox Translations : traduction automatique en local, ça me sert parfois
- Google Container : ouvre automatiquement les sites google dans un *container* dédié
- InlineDisposition Reloaded : règle un problème de téléchargement de certains sites (ça me demandait toujours ce que je voulais faire du fichier même si le type de fichier avait un réglage permanent)
- IPvFoo : indique si le site visité est en IPv6
- LibRedirect : redirige certains sites vers des *frontends* alternatifs (ex : nitter pour twitter)
- Mastodon — Fédération simplifiée ! : redirige automatiquement vers son instance Mastodon quand on veut interagir depuis une autre instance
- Mastodon Share : pour toujours avoir un formulaire de pouet à portée de main
- Plasma Integration : intégration des notifications Firefox à KDE
- QookieFix : rajoute un bouton « tout refuser » sur les bannières RGPD qui n'en ont pas

- Redirect AMP to HTML : pour ne pas utiliser AMP, qui est une technologie caca
- SingleFile : pour enregistrer toute une page (css, js, images... inclus) dans un seul fichier, pratique pour envoyer un article à quelqu'un
- snoozetabs : mettre des onglets en sommeil
- Stylus : ajouter des règles CSS aux sites
- uBlock Origin : bloquer les pubs et autres traqueurs
- uMatrix : avoir un contrôle fin sur les serveurs contactés depuis une page
- Wallabager : intégration avec mon [Wallabag](#)
- WAVE Evaluation Tool : outil de test d'accessibilité

# Astuces

## Navigation au clavier

En tapant  (l'apostrophe non typographique) en dehors d'un champ éditable sur une page, vous pouvez faire une recherche texte qui ne concerne que les liens.

En tapant sur la touche Entrée, vous activerez le lien sélectionné. Vous pouvez aussi utiliser la touche Tabulation : le focus ayant été mis sur le lien sélectionné, vous vous déplacerez à partir de celui-ci.



Divers

# Pour les screencasts : Vokoscreen-ng

J'ai utilisé plusieurs logiciels de [screencast](#) (captures vidéos de l'écran) au fil des années mais je crois que j'ai trouvé la pépite qu'il me fallait : [Vokoscreen-ng](#)

Une indication claire de la zone capturée, des raccourcis claviers pour lancer et arrêter la capture et même une mise en évidence de la souris !

Juste ce qu'il me fallait pour les 2 ou 3 screencasts que je fais par an.

# Pour surveiller des pages web : urlwatch

Si je veux surveiller l'évolution d'une page web qui ne propose pas de flux de syndication (quelle honte !) comme par exemple [la page des releases de limesurvey](#), j'utilise [urlwatch](#).

C'est pas joli, c'est du CLI, mais ce qui est formidable, c'est que c'est très flexible (on peut enchaîner des modificateurs, genre `sed`, `grep`, etc.) et que ça peut se mettre en cron sur un serveur avant de s'oublier : je recevrais alors les notifications de modifications par mail.

C'est low-tech, ça va à l'essentiel, j'adore.

Divers

# Mon shell : zsh

J'ai choisi [zsh](#) comme *shell*, histoire d'avoir un truc avec plus de fonctions que `bash` sans pour autant avoir un truc trop peu connu.

En effet, divers outils proposent des fichiers d'autocomplétion. Pour `bash` en premier lieu et s'ils supportent d'autres *shell*, zsh sera le deuxième sur la liste.

## Thème

Pour avoir un thème sympa avec un *prompt* qui me donne un certain nombre d'informations, j'utilise [powerlevel10k](#).

## Gestion des virtualenv Python

Pour charger et décharger automatiquement les `virtualenv` Python, j'utilise [zsh-autoswitch-virtualenv](#), en le modifiant un peu pour personnaliser le nom des dossiers contenant les `virtualenv`.