V.1.0 от 05.05.2016
Стуленков И.Н.
http://istulenkov.ru/personal_management_system_concept
ссылка на репозиторий github с кодом

Идея:

Современное развитие технологий движется в сторону предсказания поведения человека, на основании отслеживания всего массива его активностей в виртуальной и физической реальности (big data).

В этом есть позитивный момент т.к. если вы видите куда человек идет, то можно ему показать наиболее короткий и эффективный путь, но, по закону единства и борьбы противоположностей, здесь возникает следующая угроза (для маркетологов наоборот перспектива), как я её вижу, это возможность не предсказывать поведение, а формировать предсказуемое поведение.

Безусловно, если смотреть еще шире, то само общество, с помощью культуры и других социальных инструментов формирует предсказуемое поведение личности, но это слишком широко т.к. культура даёт несравнимо больший уровень свободы суждений в отличие от фокусированной информационной ленты, поэтому сужаюсь обратно.

Итак, что может сделать отдельный customer, чтобы усложнить задачу по формированию собственного предсказуемого поведения? Для меня ответ заключается в следующей фразе - чтобы научиться чем-то управлять необходимо научиться это измерять.

Но, что может измерять человек сам у себя и как это анализировать? Решение придумано довольно давно - это ведение личного дневника (своямоя little data) и последующий анализ зафиксированных событий.

Но при ведении дневника возникает следующая сложность, по крайней мере для меня, это почти невозможность эффективной категоризации событий\объектов (я порочен объектно-ориентированным мышлением), например, хотелось бы отдельно рассматривать задачи, мысли, вопросы, ошибки, и т.д., видеть связи между ними, получать аналитику. Надо отметить, что, частично, примеры эффективного решения данной проблемы можно найти в многочисленных книгах по тайм-менеджменту. Но я попробовал сделать свой велосипед.

Итак, какими свойствами может обладать идеальный дневник:

- Заведение данных и их быстрая (в пределе, автоматическая) классификация, по разным видам объектов, таких как, мысли, задачи, напоминания, контакты, сон, заметки и т.д. Ведение связей между объектами.
- Загрузка данных из внешних источников (соц.сетей, банковских счетов, трекеров физ.активности).
- Автоматическое построение аналитики, по разным срезам информации.
- Выдача подсказок по текущим действиям (задача с высоким приоритетом, дни рождения и т.д. и т.п.).
- Возможность настройки алгоритмов, по индивидуальной обработке объектов.
- Система должна предоставлять выбор, какая информация доступна для публикация во внешнюю сеть, а какая остается персональной (реализация должна обеспечивать честность в этом отношении), данные во внешней сети также классифицируются и раскладываются на составляющие.
- В пределе любая информация, с которой сталкивается человек, и любая информация, которая генерится данным человеком должна классифицироваться и укладываться в такой дневник.

А теперь закончив с мечтами возвращаемся из платоновского идеального в наше грешное и пропустив через множество кривых зеркал, и не только зеркал :(получаем реализацию концепта см. видео.

Вставка видео

Если интересна техническая реализация, то прошу под кат

Задача:

Изначально я ставил перед собой следующие цели:

1. Создание любых видов объектов без разработки.
2. Вывод одинаковых объектов на отдельной странице с отображением в табличном виде, а также с возможностью перехода к отдельному объекту и отображения\ведения в виде карточки.
3. Возможность установления связей между объектами.
4. Настройка статусов и категорий для объектов.
5. Ведение аналитики\напоминалок\прогнозирования

Результаты:

Получилось:

- Создание новых объектов
- Создание статусных схем, привязываемых к любым видам объектов.
- Есть возможность создавать ссылки между объектами, ссылки отображаются в отдельной вкладке. По сути ссылки решают задачу наследования и являются связями между любыми объектами.
- Гиперссылки на объекты - Возможно задавать прямую ссылку на объект.
- Для задач есть возможность создания отдельных папок для удобства обработки.

Проблемы:

- Нет аналитики.
- Нет каскадного удаления дочерних объектов
- Нет возможности поиска объектов
- Нет возможности на одной странице обрабатывать несколько объектов
- Сложность создания новых видов объектов (описание объекта занимает порядка получаса, даже у меня как автора)
- Более менее понятная архитектура превратилась в набор хаков (для новой версии будет необходимо переписывать заново).

Решение:

Disclaimer! Автор не является профессиональным программистом и специалистом в области разработки на python\js, а скорее продвинутым любителем, поэтому идеи по правильной организации архитектуры и используемым технологиям приветствуются.

Таблица 1 Используемый инструментарий

Backend	
Python 3.2.5	Серверная часть
SQLite 3.5.1	База данных
Frontend	
Bootstrap 3.3.6	Отображение данных
JQuery 2.2.0	Логика
TinyMCE 4.3.2	Редактор текста. Дополнительно использовался плагин syntaxhighlighter, для форматирования кода.
Datatables 1.10.10	Отображение данных в табличной форме.

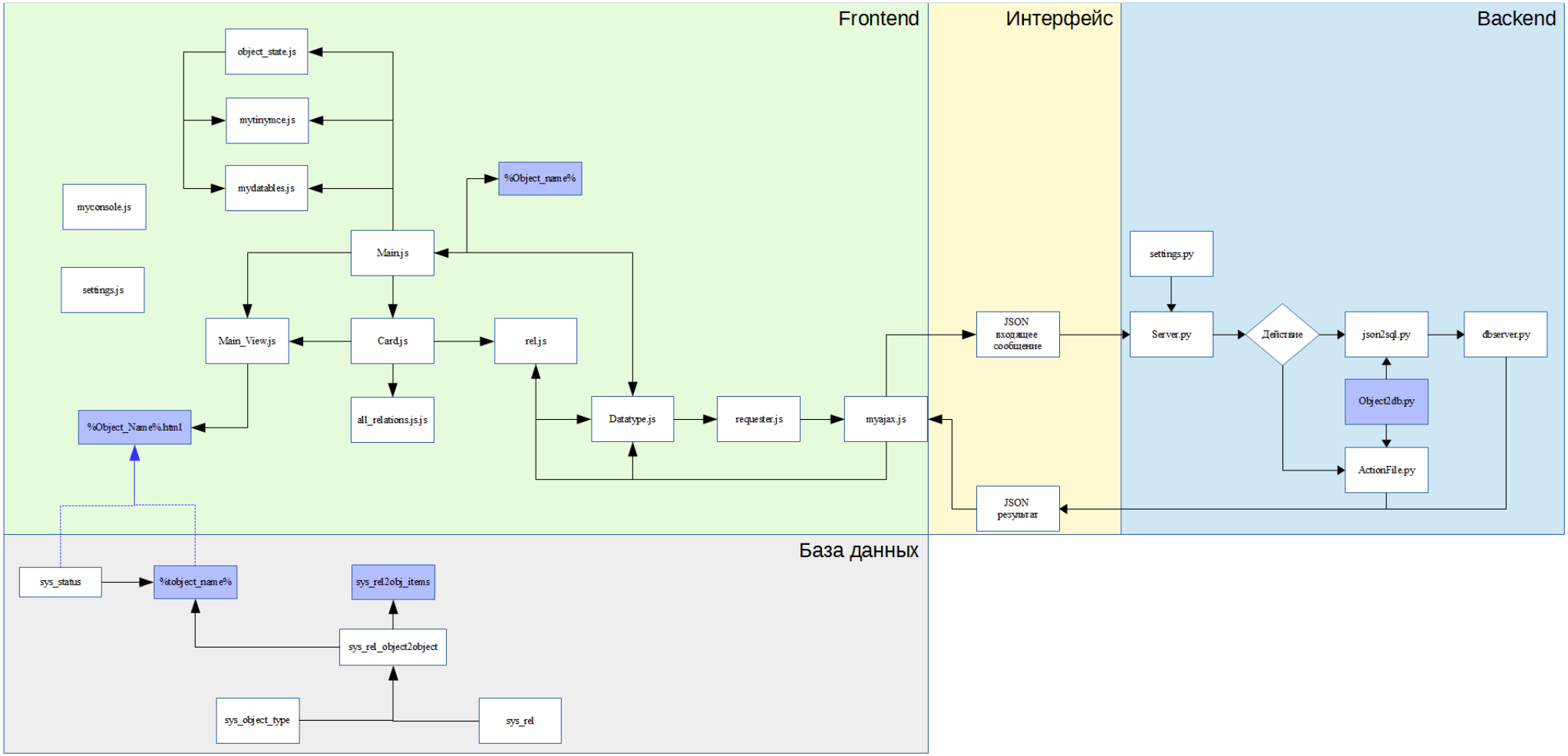

Рисунок 1 Архитектура решения

Общее описание архитектуры:

1. На Python написан сервис, который крутится на localhost и выполняет обработку запросов от html-страниц.
2. Объекты представляют собой html-страницы с определенной разметкой, в которой указываются названия полей, соответствуют полям в базе данных (не используется никаких ORM решений, обработка осуществляется своим велосипедом)
3. Общение с сервером происходит через json-сообщения.
4. Все объекты с сервера загружаются при первоначальной загрузке страницы, в дальнейшем происходит обмен только дельтой.
5. Есть встроенные объекты, это статусы и отношения, остальные объекты описываются отдельно (см. ниже)

К сожалению, по мере нарастания функционала и желания поскорее получить законченное решение, архитектура начала расползаться и не удалось в полной мере инкапсулировать объекты, поэтому на практике наличествуют разные хаки и моменты пробития инкупсуляции, но в целом схема актуальна.

Frontend	
Main.js	Точка входа для обработки событий frontend. Производится инициализация начальных данных, обработка событий с html-страницы, вызов других объектов для работы.
datatype.js	Внутреннее преобразования данных в необходимый формат для обработки в зависимости от сценария.
card.js	Обработка карточки объекта, отображение данных карточки для редактирования, просмотра, связей, сохранение данных.
main_view.js	Управление отображением карточки\таблицы на странице
myajax.js	Обертка для ajax-запроса
mydatatables.js	Обертка для работы с datatables.net.
mytinymce.js	Обертка для работы с TinyMCE
%ObjectName%	Параметры, по каждому виду объекта.
all_relations.js	Все возможные виды объектов, доступных при отображении отношений.
object_state.js	Класс для обработки вида объекта
rel.js	Обработка отношений между объектами.
requester.js	Преобразование данных в формат для передачи на сервер.
settings.js	Константы.
Backend	
server.py	Обработка запросов.
json2sql.py	Преобразование json-запроса с frontend в sql-запрос для БД
dbserver.py	Выполнение sql-запроса.
settings.py	Параметры.
object2db.py	Параметры видов объекта на сервере.
actionfile.py	Создание директорий для задач.
Схема базы данных	
%object_name%	Отдельная таблица для каждого создаваемого объекта.
sys_status	Возможные статусы, сгруппированные по статусным схемам.
Sys_rel2obj_items	Данные по связям между объектами

Sys_rel_object2object	Настройка соответствия отношений между объектами
Sys_object_type	Описание типов
Sys_rel	Описание отношений
Примеры JSON-сообщений	
Получить статусы по объекту	{"object": "status", "action":"select", "items":[{"id":1,"msg":{"fields": [{"column":"id", "op":">=", "value":"0"}]}}]}
Получить отношения по объекту	{"object": "rel_items" , "action":"select", "items":[{"id":1,"msg":{"fields": [{"column":"obj1_type", "op":"=", "value":"project"},{"column":"id1", "op":"=", "value":"22"}]}}]}
Получить возможные отношения объектов	{"object": "sys_rel_object2object", "action":"select", "items":[{"id":1,"msg":{"fields": [{"column":"id", "op":">=", "value":"0"}]}}]}
Получить все объекты	{"object": "project", "action":"select", "items":[{"id":1,"msg":{"fields": [{"column":"id", "op":">=", "value":"0"}]}}]}
Создать объект	{"object":"project", "action":"insert", "items":[{"id":0,"msg":{"fields":{ "title":"123","note":" 123 " }}}]}
Изменить объект	{"object":"project","action":"update","items":[{"id":0, "msg":{"fields_set":{"title":"1234","note":" 123 "}, "fields_where":{"id":"22"}}}]}

Где:

- Object – тип объекта (например, проект, задача и т.д.)
- Action – тип действия (создание, удаление, измение, удаление, получить)
- Items – в рамках одного сообщения, можно запрашивать несколько элементов, например, делать несколько insert.
- Msg – сообщение в рамках одного пакета
- Fields – поля по данному объекту

Некоторые из созданных объектов:

- Project - ведение проектов.
- Task – ведение задач
- Question – ведение вопросов
- Think – ведение мыслей, идей и т.д.
- Timing – контроль времени ()
- Contact – ведение контактов
- Diary – ведение личного дневника
- Snippet – snippet

Алгоритм создания нового объекта:

1. Создать таблицу в БД
2. Создать файл %object_name%.html
3. Создать файл %object_name%.js
4. Описать объект в файле object2db.py
5. При необходимости добавления статуса заполнить данные в таблицах (sys_status)
6. При необходимости создания новых отношений заполнить данные в таблицах(sys_rel_object2object)

Более подробно не описывал создание (и не записывал видео, хотя изначально идея была), т.к. получился довольно долгий и нудный процесс. Не стал делать конструктор, т.к. для концепта он оказался избыточным, но это первое с чего начну при разработке версии 2.0 см. ниже

Планы на будущее для версии 2.0:

Идеал – работа с любыми объектами в рамках одного окна, минимум действий с помощью мышки. Максимум перевести на описание в командной строке.

Идея	Описание
Редактор объектов.	Создание объектов, без необходимости ручного создания html-страниц, наличия отдельного js-файла для объекта и отдельной настройке на python. Создание статусов. Создание\настройка отношений.
Отображение на одной странице множества объектов.	В настоящий момент архитектура построена таким образом, что на одной странице возможно отображение только одного вида объекта (не считая ссылок на другие объекты).
Командная строка	Командная строка для сокращения кол-ва действий для выполнения операций. Например, находясь на объекте «Проект» в командной строке набирается «Задача» и происходит автоматическое создание объекта «Задача» со ссылкой на данный проект. Т.к. в настоящий момент нужно перейти на вкладку
Шифрование.	Добавить шифрование БД.
Аналитика	Возможность базовой аналитики по объектам, с отображением кол-ва по статусам, важности и т.д.
Добавление объекта сроки(даты)	Отдельный объект, который можно добавлять в любой объект с возможностью добавления стоков
Активности	Возможность отработки определенных действий
Проверка на дублирования	При создании инстанции объекта проверка на дублирование
Обязательность полей	Определение полей, обязательных для заполнения.

Заключение:

Старался написать статью максимально короткой, чтобы быстро можно было понять основную идею и описание реализации, не утонув в большом количестве лишней информации. В случае, если у сообщества возникнет желание прочитать подробней про какие-то моменты, то напишу дополнительный пост.

Спасибо за уделённое время.