Тестовое задание:

1. Создать Django приложение в котором можно добавлять URL's в административном разделе (/admin/). С возможностью указать сдвиг во времени (минуты, секунды) через сколько какой URL когда будет обработан (timeshift).
   - Frontend
     - Сделать два окна textarea.
     - В первое окно выводить информацию об успехе или ошибке обработки URL'а (для backend можно воспользоваться очередями Queue).
     - Во второе вывести данные полученные из пункта 2.

2. Создать сервер для парсинга сайтов по URL'ам указанным в базе Django приложения (п. 1), в указанный сдвиг времени или сразу запускать если не указан сдвиг.
   - На сайтах получить
     - Заголовок (title)
     - Определить кодировку страницы
     - Если есть, найти и получить H1
   - Вывести данные во второе окно.

Основные моменты:

I. Сервер для обработки URL может быть выполнен в виде команды, либо в виде полноценного сервера с собственными процессами/нитями (process/thread).
II. Данные по обработанным URL'ам сохранять в базу и при перезагрузке выводить их во втором textarea.
III. Все сообщения в первом окне должны начинаться с даты и двоеточия "**<дата dd.mm.yyyy HH:mm:ss>:** ".
IV. Все сообщения во втором окне начинаются с URL и тире (dash), вида "http://example.com - ", с последующим перечислением что получено и чего нет. Если получить ничего не удалось - тогда просто URL.
V. Запрещается использовать любые брокеры сообщений которым нужны собственные серверные платформы (RabbitMQ, ActiveMQ и т.д.)
VI. Рабочий сайт должен запускаться из командной строки **одной командой**, вида "**python manage.py runserver**" или "**python run.py**" или "**python -m module -c command**", либо bash/sh скриптом.
VII. Все зависимости должны устанавливаться из файла "requirements.txt", который будет лежать в корне кодового репозитория (зависимости которых **нет в Pypi,** указывать в "http[s]://" формате).
VIII. Описать процесс запуска кода на локальной машине
IX. Код должен запустится и работать на машине на которой установлен только python и сопутствующие библиотеки, SQLite.

Плюсом рассматривается:
- использование базовых модулей Python (из комплекта поставки)
- использование на frontend'е Websockets.
- Обвязка (wrapping) стандартного Django WSGI Сервера методами кот. могут в мультипроцессорном потоке запускать процессы по обработке URL's (не обязательно).
- Мульти-версионный и переносимый код (Python 2.5,2.6,2.7,3.2,3.3,3.4).

Репозитарий опубликовать в любом доступном месте (GIT, Mercurial, Bazaar etc.) как публичный.

github.com, gitlab.com, bitbucket.org, Sourceforge (sf.net), launchpad.net

3. Написать SQL запрос в базу (один запрос) который выберет данные из таблиц 1,2,3 и запишет в таблицу 4

таблица 1

ID	Name	Surname	Salary/year
1	John	Terrible	11000
2	Maggie	Woodstock	15000
3	Joel	Muegos	22000
4	Jeroen	van Kapf	44000

таблица 2

ID	Month	Taxes	EmployeeID
1	01.01.15	250	1
2	01.02.15	267	1
3	01.01.15	300	2
4	01.02.15	350	2
5	01.01.15	245	3
6	01.02.15	356	3
7	01.01.15	246	4
8	01.02.15	356	4
9	01.03.15	412	3

таблица 3

ID	InternalNumber	Position	EmployeeID
1	32894	Manager	1
2	23409	Top Manager	2
3	23908	CEO	3
4	128	Board Chairman	4

таблица 4

InternalNumber	Name/Surname	Position	Salary/Month	Tax	Month

Вывести все данные где они присутствуют для всех полей таблицы 4