## Домашняя работа к уроку 8.

Студент: В.Мокренко.

## Антипаттерны в коде

### Magic Number // Магические числа

*Вызовов функций с параметрами-числами нет в программном коде*

### Spaghetti Code // Спагетти-код

*Анализ кода показал, что существуют места с избыточной сложностью, потенциально требующие рефакторинга и упрощения. Я бы выделил три места:*

*Модуль tasks содержит основные классы домена проблемной области. В классах есть избыточность по определению атрибутов по умолчанию*

*Модуль mapregistry содержит класс DataMapper, который требуется дополнительных комментариев в коде, т.к. понимание использования методов класса не самое очевидное.*

*Файлы с настройками стилей css имеют избыточность и требует упрощения.*

### Lasagna Code // Лазанья-код

*Есть понимание, что требуется более внимательно подойти к описанию классов проблемной области – tasks, tasksScheduled, tasksHeaped. Есть много одинаковой функциональности и много отличной друг от друга. Изменения по урокам привели к некоторой многослойности в описании классов. Требуется более четкое понимание какие атрибуты надо наследовать, какие однозначно -нет, должен ли быть в каждом классе свой искусственный ключ или они должны разделять общий.*

### Blind faith // Слепая вера

Особых правил корректности ввода в коде нет, но, однозначно, в хорошем приложении необходим отдельный блок валидации плюс использование визуальных контролов, обеспечивающих валидный ввод . Блок валидации должен обеспечивать также проверку на ограничения длины и типы полей БД.

### Cryptic Code // Шифрокод

Писать шифрокод – не уважать свой труд)).

### Hard Code // Жёсткое кодирование

1. Есть несколько мест, где прописаны конкретные имена, например имя логгера, имя файла. Предпочтительным видится сделать один глобальный configmap и, при необходимости, локальные, в которые вынести абсолютно все захардкоженные имена.
2. Есть типы полей БД с указанием конкретной длины . Возможно требуется пересмотреть типы на возможность исключения ограничений длины, например, использовать просто тип varchar.

### Soft Code // Мягкое кодирование

Реализация задачи не налагает особых требований на пользователя системы и не требует какого-то сложного конфигурирования. Структура модулей сделана так, чтобы не вызывать сложности в небольших изменениях. Но несколько настраиваемых элементов я бы добавил, например, кол-во одновременно отображаемых строк на экранах с табличными данными.

### Lava flow // Поток лавы

После выполнения каждого ДЗ код тестирвался полностью, в том числе не задеваемый функционал, поэтому серьезных проблем по данному антипаттерну, предполагаю нет. Но т.к. зависимостей между модулями достаточно много и функционал менялся (например, работа без CBV, а потом с CBV),то нельзя исключать наличие проблемы. Требуется оптимизация кода для лучшей структурированности модулей/классов и выбора самых оптимальных вариантов реализации.

## Антипаттерны ООП

### Anemic Domain Model // Боязнь размещать логику в объектах предметной области

*Я много работал с базами данных, поэтому для меня естественно не бояться добавлять логику в объекты предметной области))*.

### God object (The Blob) // Божественный объект

Такого типа объект только один – engine, содержит ядро логики работы. В моем случае он очень простой. Единственным моментом, который я бы изменил, это сделать более тонкие контроллеры. Есть некоторая перегруженность модуля views.py. И, наверное, модуль tasks надо упростить, чтобы разделить атрибуты и методы предка, которые можно стандартизировать для их использования в потомках.

### Poltergeist // Полтергейст

*Очень надеюсь, что таких нет, т.к. достаточно много внимания уделялось тестированию сайта.*

### Singletonitis // Сплошное одиночество

Возможные проблемы очевидны, поэтому в моем коде этого нет.

### Privatization // Приватизация

Не используются

## Методологические антипаттерны

### Copy — Paste // Программирование методом копирования — вставки

Больших кусков, сделанных через copy/paste и требующих какой-то унификации из-за этого, не нашел.

### Golden hammer // Золотой молоток

Даже близко нет таких классов или модулей в приложении.

### Improbability factor // Фактор невероятности

Есть несколько методов, в которые лучше добавить блок exception.

### Premature optimization // Преждевременная оптимизация

Оптимизация не производилась.

### Reinventing the wheel // Изобретение велосипеда

Проект учебный. Многое написано с нуля. Но, например, верстка взята как готовая часть с freeware-ресурса.

### Reinventing the square wheel // Изобретение квадратного колеса

Сделан свой ORM, а не взят какой-то готовый с целью понимания процессов и паттернов.

## Архитектурные антипаттерны

### Abstract Inversion // Инверсия абстракции

В коде нет методов, где функциональность пользователю не нужна и поэтому ее можно скрыть. Количество методов – минимум необходимого для выполнения ДЗ и обеспечения необходимой функциональности.

### Big ball of mud // Большой комок грязи

Однозначно проблема не стоит, т.к. код имеет структурированность и сделан на основании шаблонов проектирования.

### Input kludge // Затычка на ввод

Данный раздел не был реализован в рамках проекта.

### Magic button // Волшебная кнопка

Уровень представления не содержит никакой логики. Возможно есть некоторый изыбток логики на ровне контроллеров и ее часть можно перенести на уровень модели.

### Mutilation // Членовредительство

Нет. Используется шаблоны, единые элементы интерфейса.

### Stovepipe Enterprise // Дымоход предприятия

Проект индивидуальный, связей нет.

### Stovepipe System // Дымоход системы

Не относится к проекту.

### Jumble // Путаница

Не относится к нашей задаче