# Vortex

## Слои

1. Ядро – содержит нейтральные универсальные реализации
   1. System – опорные интерфейсы и перечисления
   2. Logger – контроллер вывода данных
   3. Settings – контроллер настроек
   4. Debug – расширение настроек для дебага
   5. App – контроллер приложения
   6. Database – контроллер базы данных
   7. Loader – система асинхронной загрузки данных
2. Реализация под движок – содержит драйверы адаптированные под платформу (в данном случае Unity)
3. Вспомогательные компоненты под текущий движок – содержит набор полезных скриптов, выполняющих стандартные действия в рамках текущего движка
   1. UI компоненты
   2. Расширения и синтаксический сахар

## Принципы построения систем (пакетов, компонентов)

### Одиночная моно-модель

* Шина доступа (именование: {System})
  + Кеш модели данных
  + Логика изменения
  + Логика вывода
  + Логика сохранения и загрузки (хранилище или сейв)
  + События изменения данных
* Модель (именование: {System}Model)
  + Паблик поля данных (get; private set;)
  + Расширения данных через партиалы
  + Функционал копирования полей по признаку на экземпляре источника (хранилища)
    - Property
    - BindingFlags.GetProperty | BindingFlags.Public | BindingFlags.Instance
    - !CanWrite.
* Хранилище данных (именование: {System}Storage)
  + Паблик поля данных (get;)
  + Расширения данных через партиалы

### Множественные модели (Стандартная БД)

* Шина доступа (именование: {System})
  + Индекс-реестр
  + Логика получения единицы
  + Логика изменения (расширения модели)
  + Логика вывода (расширение модели)
  + Логика сохранения и загрузки (хранилище или сейв)
* Модель (именование: {System}Model)
  + Паблик поля данных (get; private set;)
  + События изменения данных
  + Интернал метод вызова событий изменения данных
  + Функционал копирования полей по признаку на экземпляре источника (хранилища)
    - Property
    - BindingFlags.GetProperty | BindingFlags.Public | BindingFlags.Instance
    - !CanWrite
* Хранилище данных (именование: {System}Storage)
  + Паблик поля данных (get;)
  + Расширения данных через партиалы
  + Метод создающий и возвращающий модель данных записи базы данных, заполненной из данного экземпляра хранилища

## Пакеты

### System

Пакет содержит в себе базовые абстракции фреймворка и перечисления (enum).

Singleton – абстрактный класс синглтона.

SystemController – абстрактный класс основного контроллера системы (центральной шины). Имеет поле для «драйвера», который определяет реализацию для конкретного движка. Драйвер должен выставлять себя сам.

SystemModel – абстрактный класс модели системы, с функционалом автоматического подтягивания данных из связанных хранилищ

ISystemDriver – интерфейс драйвера контроллера системы. При назначении драйвера автоматический будет вызвана функция Init, при отключении - Destroy

### Settings

Пакет содержит базовую систему настроек. Модель расширяется через партиалы, хранилище расширяется через наследование от абстракции. SystemModel поддерживает сбор данных из нескольких объектов-хранилищ в одну модель.

Для Unity используется реализация хранения в ScriptableObjects с одним родителем и сбор этих ресурсов через Resources.LoadAll<T>() с дальнейшим сбором данных в модель через Model.CopyFrom(scriptableObject).

### Debug

Пакет содержит расширение системы настроек для ввода флагов дебаг-режима.

### App

Пакет содержит основную модель данных приложения и ее контроллер. Через нее идет управление и считывание текущего состояния приложения. Фактически является машиной состояний приложения. Может быть расширена при необходимости через партиалы. В реализации для Unity создается FocusHandler, который управляет состоянием приложения, считывая свое состояние как Monobehavior.

### Loader

Пакет содержит систему асинхронной загрузки данных. Наследники интерфейса ILoadable могут зарегистрироваться в очереди на загрузку. На данный момент система срабатывает один раз при запуске приложения. Нужно доработать для подгрузки данных в рантайме.

### Logger

Пакет содержит систему вывода данных в консоль. Драйвер для Unity производит вывод через Debug.Log(Warning, Error).