**Документация для фреймворка.**

***Техническая реализация:***

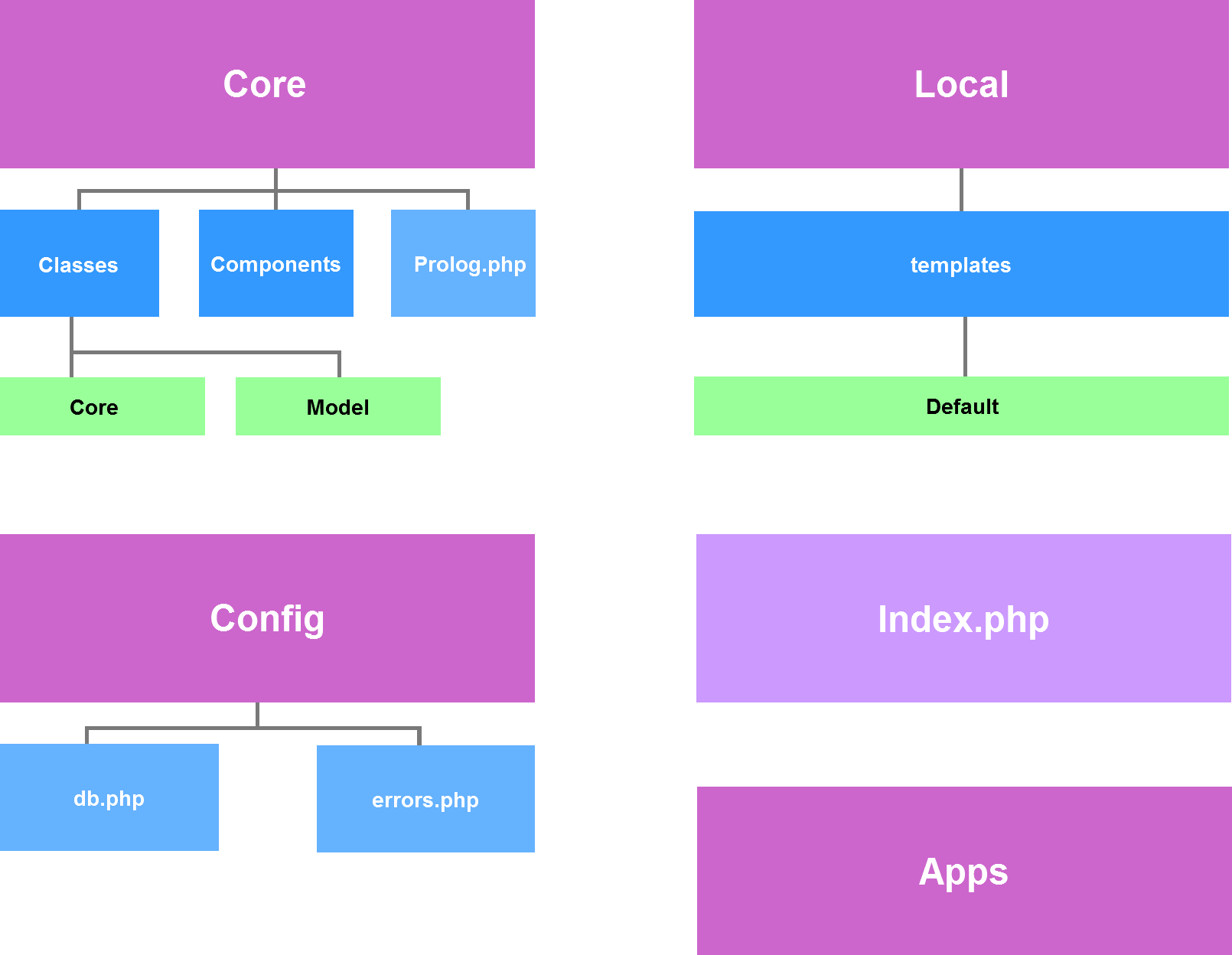
Данный фреймворк функционально повторяет набор классов Bitrix framework, но с отличиями в архитектуре и без графического административного интерфейса.

Bitrix framework построен на паттернах проектирования **Observer**, **Singleton**, **MVC**, **Adapter**, **Decorator** (всего понемногу).

В своей реализации фреймворка, я решил использовать шаблоны

**Front controller, MVC,** а так же, я решил перенять некоторые другие их решения.

**Структура папок:**



**Описание папок:**

1. **Core** – папка ядра. Она содержит файлы ядра и состоит из следующих папок:
   1. **Classes** – папка с классами, организующими работу фреймворка.
      1. **Core** – классы, обеспечивающие работу фреймворка (маршруты, компоненты, ошибки, шаблоны и т.д.)
      2. **Model** – классы, каждый из которых соответствует сущности приложения (подарки, пользователи и т.д.)
   2. **Components** – папка с компонентами. Компонент – контроллер, работающий с моделью и генерирующий шаблон.
   3. **Prolog.php** – инициализирует загрузчик классов и начинает сессию.
2. **Apps** – папка разделов сайта. После того, как ЧПУ (class Route) получит запрос, обработает его и решит, что нужно подключать, он берёт нужный файл из этой папки и подключает его как тело страницы.
3. **Config** – папка с конфигами сайта (данные подключения к БД, коды и пояснения ошибок и т.д.)

**3**.1) db.php – хранилище данных подключения к БД.

**3**.2) errors.php – хранилище кодов ошибок и пояснений к ним.

1. **Local** – хранилище шаблонов сайта, картинок, стилей, js и т.д.

**Структура классов:**

1. **Core\Auth** – класс, отвечающий за авторизацию, регистрацию и проверку сессии пользователя**.**
   1. **Core\Auth::** **passwordValidation()** – метод, проверяющий валидность введённого пароля.
   2. **Core\Auth::** **emailValidation()** – метод, проверяющий валидность введённого логина.
   3. **Core\Auth::** **randomPasswordGenerator()** – метод, генерирующий рандомный пароль.
   4. **Core\Auth::** **checkLogin()** – метод, проверяющий наличие пользователя по его логину.
   5. **Core\Auth::** **checkPass()** – метод, проверяющий правильность пароля (при авторизации).
   6. **Core\Auth::** **authStart()** – метод, начинающий сессию пользователя.
   7. **Core\Auth::** **authClose()** – метод, завершающий сессию пользователя.
   8. **Core\Auth::** **authCheck()** – метод, проверяющий сессию пользователя.
   9. **Core\Auth::** **restorePassword()** – метод, восстанавливающий пароль пользователя (не реализован ещё).
   10. **Core\Auth::** **confirmEmail()** – метод, подтверждающий Email пользователя (не реализован ещё).
2. **Core\includeComponent** - класс, отвечающий за реализацию паттерна MVC и реализующий функционал компонентов.

**2.1) Core\includeComponent::** **includeComponentCode()** – закрытый метод, подключающий тело компонента (контроллер).

**2.2) Core\includeComponent::** **includeComponentTemplate()** – метод, подключающий шаблон компонента.

**2.3) Core\includeComponent::** **add()** – метод, вызывающий компонент и передающий ему параметры.

1. **Core\Config** – класс, отвечающий за быстрый доступ к конфигурационным данным (БД, коды ошибок и т.д.)

**3.1) Core\config::** **getAllDbInfo()** – метод, возвращающий параметры подключения к базе данных.

**3.2) Core\config::** **getDbHost()** – метод, возвращающий хост базы данных.

**3.3) Core\config::** **getDbUsername()** – метод, возвращающий логин пользователя БД.

**3.4) Core\config::** **getDbName()** – метод, возвращающий имя базы данных.

**3.5) Core\config::** **getDbPass()** – метод, возвращающий пароль пользователя базы данных.

**3.6) Core\config::** **getError()** – метод, возвращающий описание ошибки по её коду.

1. **Core\Database** – класс, отвечающий за работу с базой данных.

**4.1) Core\database::** **startConnect()** – метод, начинающий сеанс работы с базой данных.

**4.2) Core\database::** **closeConnect()** – метод, завершающий сеанс работы с базой данных (не реализован ещё).

1. **Core\includer** – класс, отвечающий за быстрое подключение файлов (картинки, JS, Css)

**5.1) Core\includer::** **checkUrlSlashInBegin()** – приватный метод, добавляющий слеш к адресу подключаемого файла.

**5.2) Core\includer::** **getJS()** – метод, генерирующий ссылку на JS файл.

**5.3) Core\includer::** **getCSS()** – метод, генерирующий ссылку на CSS файл.

**5.4) Core\includer::** **getIMG()** – метод, генерирующий ссылку на картинку.

1. **Core\Route** – класс, отвечающий за маршрутизации в приложении.

**6.4) Core\Route::** **startRoutes()** – метод, управляющий маршрутами.

1. **Core\Templates** – класс, отвечающий за подключение шаблонов страниц и компонентов.

**7.1) Core\templates::** **getTemplates()** – метод, получающий список шаблонов страниц.

**7.2) Core\templates::** **getCurrentTemplate()** – метод, получающий текущий шаблон данной страницы.

**7.3) Core\templates::** **getTemplatePath()** – метод, возвращающий путь к шаблону.

**7.4) Core\templates::** **getCurTemplatePath()** – метод, возвращающий путь к текущему шаблону.

**7.5) Core\templates::** **getHeader()** – метод, возвращающий путь к хедеру текущего шаблона.

**7.6) Core\templates::** **getFooter()** – метод, возвращающий путь к футеру текущего шаблона.