**ВИДЖЕТ КОТОРЫЙ ПОЛНОЦЕННЫЙ СЮЖЕТ РЕШАЕТ**: **НАПРИМЕР, ОТКРЫТИЕ И ЗАКРЫТИЕ МОДАЛЬНОГО ОКНА**  
  
Преабула:

Приложение строится аналогично ОС. У Приложения есть ядро(core) и сервисы(app).

Core предоставляет различные базовые переиспользуемые функции: UI-kit(UI), библиотеку компонентов (Components), Шаблонов(templates), и виджетов(widgets), функции для работы с сервером(api), а также вспомогательные функции (нужна доработка).

В свою очередь app непосредственно реализует функционал нашего приложения. Оно настраивает раутинг(routing) и состоит из страниц(pages). При проектировании страницы стоит использовать следующие абстракции:

1. Общие рекомендации
   1. Стили могут затрагивать или сам компонент или расположение его дочерних элементов. Завпрещена стилистичесская настройка потомков! Каждый компонент по сути изолируется. Его стили не должны влиять ни на его родителя, ни на потомков(кроме дочерних эелементов и у тех только компоновка относительно дроуг друга)
   2. UI-kit  
      Если состоит из нескольких тегов то нужно возвращать его обернутым **в div?**Если необходима его внешняя кастомизация использовать передачу дополнительных стилей через пропс**, приэтом нельзя, чтобы стили влияли на внутреннее устройство элемента, только на его расположение относительно других элементов  
      *возможно стоит использовать две версии: с оберткой и без, где при использовании версии без обертки пользователь сам должен обернуть в контейнер полученный компонент***
   3. Определен следующие зависимости  
      страница->виджеты->кмопоненты->UI-kit  
      Контейнер влияет исключительно на расположение элементов
   4. Виджет  
      Виджет содержит два аргумента (конфигурация и синхронизации). При написания виджета необходимо ответить на следующий вопрос: какую конкретную бизнес задачу он решает и какую синхронизацию он должен выполнить. Так как ***один виджет – одна бизнес задача*** синхронизационная функция одна(случае использования useReducer – это колбек который вызывает диспатч с одним из действий).   
      **Виджет сам встраивает синхронизационную функцию в его компоненты или UI  
      РЕШЕНИЕ БИЗНЕС ЗАДАЧИ ВИДЖЕТ КОНФИГУРИРУЕТ САМ, ЕДИНСТВЕННОЕ ЧТО ПРИХОДИТ ИЗ ВНЕ – ЭТО СИНХРОНИЗАЦИЯ**как понять, что виджет написан правильно с точки зрения изоляции? Должна быть возможность вставить его на страницу и задав ему конфигурацию, иметь возможность наблюдать его автономную работу.  
      Может ли кнопка быть виджетом?   
      Допустим кнопка должна отправлять данные на сервер, тогда в качестве конфига ей надо передать колбек отправки данных.   
      Может ли моя форма быть виджетом при этом не отправлять данные на сервер?  
      Да может, если бизнес задача формы – отобраажать ее ищменения и следить за своим состоянием.   
        
      **Возможно несколько синхронизаций**
2. Страница(page)
   1. Страница нашего приложения, которое реализует некий функционал, решающий несколько бизнес-задач
   2. Зависимости:  
      Виджеты(Widgets) – Внутренние(предпочтительно) или из Core(не рекомендуется)  
      Компоненты(Components) – Вну
   3. Рекомендации:  
      Использовать внутренние виджеты и UI-kit. Не рекомендуется использование на верхнем уровне компонентов, т.к. Они сложно управляемы.  
      Страница выступает в роли “локального стейт менеджера”. Она обеспечивает синхронизацию между ее дочерними компонентами и хранит свое состояние. Передает Дополнительно функцию синхронизации
   4. Контейнеры и Шаблоны(обертки)  
      Разделяются два типа оберток(шаблоны/контейнеры): страничные(не изолированные) и **локальные(изолированные)**Локальный, как следует из названия изолирован, то есть его стили они влияют только на расположение элементов в самом компоненте. Предполагается, что вставка такого шаблона в компонент не сможет отразиться ни на родителе компонента, ни на его братьях/сестрах.   
      Пример: обертка, которая принимает две комонента и задает им display: flex с flex-gap: 10px.   
      **страничные (не изолированные)**Такой шаблон напротив при использовании может повлиять на соседей/родителя.  
      Пример: обертка для всплывающего окна, которая растягивается на всю страницу(через position: fixed). При попытке расположить такой эелемнт внутри какого-то компонента или виджета, он будет влиять на соседей(его поведение не изолировано)
      1. Шаблоны(templates)  
         Поскольку контейнер это шаблон, то есть предполагается что он может иметь строго определенную конфигурацию(присутствуют такие то и такие то элементы, не больше не меньше), в нем настраиваются экстерналии элементов которые он будет в себе располагать через className.nth-child()
      2. Контейнеры(Contsiners)  
         Элемент может сам настраивать свои экстерналии для данного контейнера. У каждого элемента, расположение которого планируется в контейнере, должны быть соответствующие доп стили для данного контейнера. Использоваться эти стили должны при передаче соответсвующего литерала в качестве пропса containerStyle  
           
         *Помимо прочего обертки разделяются на шаблоны и контейнеры.  
         Если элемент монтируется напрямую в страницу, то его экстерналии(внешние стили, которые могут повлиять на другие элементы(ширина,высота, расположение и тд)) передаются ему в качестве пропса outerStyles.   
         Если же элемент будет размещаться на странице в контейнере(templates), допустимы несколько вариантов:*

Про устройство APP

# APP

Само приложение, в отличие от core, оно уже неспоредственно решает поставленную проблему, если core – это библиотека, то вот app уже непосредственно представляет собой приложение

APP состоит из нескольких файлов/директорий:

1. Main.tsx  
   Входная точка в проект, которая подключает Tanstack query, React Router.  
   *В будущем может подключать и глобальный стейт менеджер приложения***main.tsx не лежит в папке app, однако поскольку это входная точка в проект, необходимо о ней было обмолвиться**
2. router/  
   Директория, которая отвечает за настройку раутинга в приложении, **также по сути дела нарушает правило импортов модулей**, однако это сделано для удобства, чтобы все страницы можно было хранить в отдельной папке, а раутинг настравать отдельно.   
   По сути дела router подключает к приложению страницы(модули)  
   1. layout.tsx  
      Базовый лейаут приложения
   2. layout.module.css  
      Стили для лейаута  
      **По сути layout.module.css нарушает принцип того, что блок не должен содержать размеров и тд, однако тк это глобальный лейаут, это исключение также допустимо**
   3. index.tsx  
      входная точка в модуль
   4. Header  
      Дочерний модуль, реализующий хэдер страницы. Пока что остался как легаси  
        
      **Одна из серьезнейших проблем, что пока что элементы лейаута не могут общаться со страницей. Как ее решить пока не знаю, не думал**
3. pages  
   Модуль в котором реализованы различные страницы приложения, каждая страница – это отдельный модуль, который может содержать свои дочерние модули, правило написания которых не должно противоречить правилам написния модулей

# Правила написания UI, Components, Widgets, Pages

Виды задач, решаемых элементами:

* Задача базового размещения  
  Просто разместить кирпичик, из которого строится Сайт, который максимально реиспользуемый
* Задача расположения  
  Расположение в каком-то конкретном порядке/виде/формате нескольких элементов
* Бизнес-задачи  
  Какие-либо технические задачи: общение с сервером, обработка данных, получение данных и тд

## UI

Определение: UI-kit, по сути, кастомная библиотека компонентов

Решаемые задач: Задачи базового размещения

Правила написания: не могут ни от чего зависеть.

Зависимости: -

## Components

Определение: элементы, состоящие только из UI и Wrapers. Не несут в себе никакой бизнес-логики

Решаемые задачи: Задачи расположения

Правила написания: Component может иметь определенные ограничения на принимаемые данные, так как располагает элементы конкретным образом, однако не может решать никаких бизнес-задач, исключителльно размещение в каком-то порядке нескольких UI

Зависимости: UI, Wrapers

Особенности:

В связи с вышеуказанным перенос components в core сложен и рискован, тк порождает горизонтальные зависимости в core

## Widgets

Определение: элемент, решающий конкретную бизнес-задачу и задачу расположения, имеющий механизмы синхронизации.

Решаемые задачи: Задачи расположения и Бизнес-задачи

Правила написания:

Widgets принимает 4 параметра:

Config – данные которые виджет должен отобразить, этот конфиг должен позволять использовать UI или Components

Sync