ЭТАП: ПРОЕКТИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

ПОСТАНОВКА ЗАДАЧИ. АРХИТЕКТУРНЫЕ РЕШЕНИЯ. ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Полезные ссылки

1. ГОСТ 19.102-77 «Стадии разработки»: *https://www.swrit.ru/doc/espd/19.102-77.pdf*
2. *Г. Буч, Дж. Рамбо, И. Якобсон. Язык UML. Руководство пользователя.*
3. *Э.Гамма, Р.Хелм, Р.Джонсон, Дж.Влиссидес. Паттерны объектно-ориентированного проектирования.*
4. Интернет-ресурс о паттернах проектирования: *https://refactoring.guru/ru*.
5. Полезный материал о диаграммах пакетов, компонентов и размещения: *https://itteach.ru/rational-rose/diagrammi-paketov-komponentov-i-razmescheniya*.
6. Материал по разработке пользовательского интерфейса: *https://livetyping.com/ru/blog/chto-takoe-razrabotka-polzovatelskogo-interfeisa-i-zachem-tt-zakazyvat*
7. Принципы разработки пользовательского интерфейса: *https://clck.ru/DcKAK*
8. Этапы разработки пользовательского интерфейса: *https://vc.ru/design/58502-etapy-razrabotki-polzovatelskogo-interfeysa-kak-sdelat-tak-chtoby-ui-ne-lishil-vas-pribyli*
9. Примеры UML диаграмм: *https://www.uml-diagrams.org/*

Краткие теоретические сведения и методические указания к выполнению лабораторной работы №4

Основная цель постановки задачи – указать назначение программы и какие цели перед ней ставятся (таблица 1). Цель, в свою очередь, показывает, что заказчик хочет видеть на выходе.

Таблица 1 – Описание назначения и целей создания программы

ПОСТАНОВКА ЗАДАЧИ
Назначение:
Указать перечень объектов автоматизации, на которых предполагается использовать программу/систему, перечень автоматизируемых органов (пунктов) управления объекта автоматизации и управляемых ими объектов (здесь указать в каких подразделениях предусматривается устанавливать программу/систему и привести в разрезе подразделений перечень автоматизируемых бизнес-процессов верхнего уровня).
Цели создания:
Наименования и требуемые значения технических, технологических, производственно-экономических или других показателей объекта автоматизации, которые должны быть достигнуты в результате создания программы; критерии оценки достижения целей создания программы/системы.

Пример 1 описание постановки задачи для темы предметной области «Разработка информационно-аналитической системы «Корпоративное хранилище данных (КХД)»:

Назначение: *КХД предназначена для повышения оперативности и качества принимаемых управленческих решений сотрудниками Заказчика.*
Основным назначением КХД является автоматизация информационно-аналитической деятельности в бизнес-процессах Заказчика.
В рамках проекта автоматизируется и-формационно-аналитическая деятельность в следующих бизнес-процессах:
1. анализ финансово-хозяйственной деятельности;
2. информационная поддержка процессов бюджетирования;
3. ...

Цели создания: *КХД создается с целью:*
- обеспечения сбора и первичной обработки исходной информации, необходимой для подготовки отчетности по показателям деятельности;
- создания единой системы отчетности по показателям деятельности;
- повышения качества (полноты, точности, достоверности, своевременности, согласованности) информации;
- ...
В результате создания хранилища данных должны быть улучшены значения следующих показателей:
- время сбора и первичной обработки исходной информации;
- количество информационных систем, используемых для подготовки аналитической отчетности;
- время, затрачиваемое на информационно-аналитическую деятельность;
- ...

Пример 2 описание постановки задачи для темы предметной области «Обновление приложения Smart TV и Android TV»:

Приложение Smart и Android TV являются продуктами холдинга «Москва Медиа», через которые осуществляется дистрибуция аудио-визуального контента. Программы передач каналов Москва 24 и Москва Доверие в виде архивов, а также прямой эфир (только Москва 24) доступны в приложении.

Основная задача продуктов – информирование локальной (московской) аудитории о происходящих событиях в регионе, а также репрезентации контента, который удовлетворяет широкие потребности данной аудитории. В связи с этим необходимо обновить существующие версии приложений и улучшить их функционал. Предполагается, что обновленные версий приложений будут более востребованы для конечного пользователя и соответствовать интересам холдинга Москва Медиа.

Предполагаемое время работ по обновлению приложений – август/сентябрь 2017 года.

Предполагаемое время релиза приложений – 4 квартал 2017 года.

Описание архитектуры программного средства

Архитектура программного обеспечения – фундаментальная организация системы, воплощенная в ее компонентах, их отношения друг к другу и к окружающей среде, а также принципы, лежащие в основе ее проектирования и развития. Архитектура включает:

- выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
- соединение выбранных элементов структуры и поведения во всё более крупные системы;
- архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для того, чтобы проектируемая система отвечала различным атрибутам качества, на практике часто используют различные архитектурные шаблоны (паттерны). Наиболее распространенные архитектурные шаблоны:

- Многоуровневый шаблон;
- Клиент-серверный шаблон;
- Ведущий-ведомый;
- Каналы и фильтры;
- Шаблон посредника;
- Одноранговый шаблон;
- Шина событий;
- Модель-представление-контроллер;
- Доска;
- Интерпретатор.

Краткий обзор архитектурных шаблонов представлен по ссылке: shorturl.at/huD18.

Проектирование может осуществляться с различных точек зрения. С позиции структурной организации в рамках лабораторной работы для проектирования архитектура будут использоваться UML диаграммы классов, компонентов, развертывания и пакетов.

Диаграммы классов (рисунок 1) – это наиболее часто используемый тип диаграмм, которые создаются при моделировании объектно-ориентированных систем. Показывают набор классов, интерфейсов и кооперaций, а также их связи.

На практике диаграммы классов применяют для моделирования статического представления системы (по большей части это моделирование словаря системы, кооперaций или схем). Кроме того, диаграммы данного типа служат основой для целой группы взаимосвязанных диаграмм – диаграмм компонентов и диаграмм размещения.

Диаграммы классов важны не только для визуализации, специфицирования и документирования структурных моделей, но также для конструирования исполняемых систем посредством прямого и обратного проектирования.

> **Диаграмма классов –** это диаграмма, которая показывает набор классов, интерфейсов, кооперaций и их связи. Графически представляет собой набор вершин и связывающих их дуг.

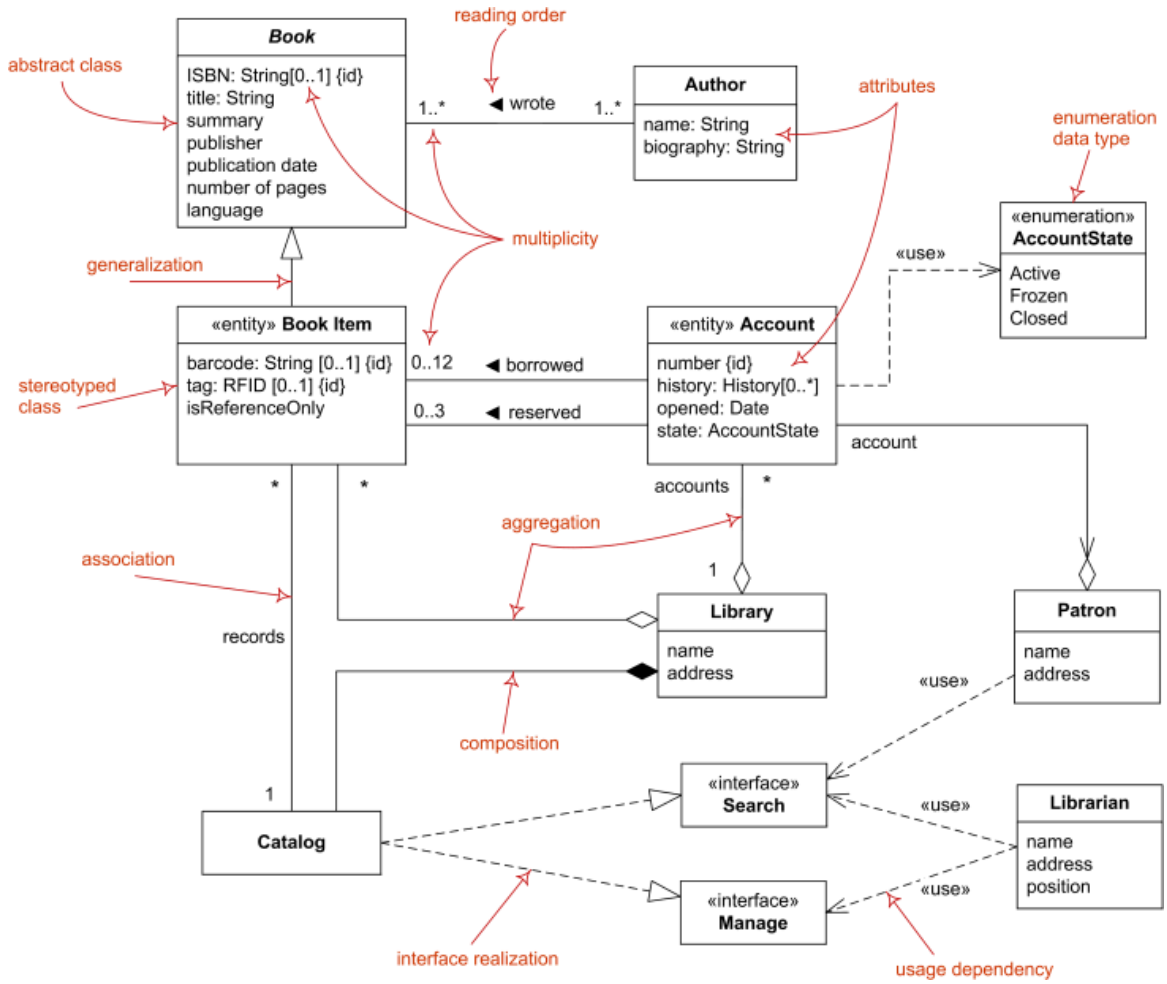

Рисунок 1 – Пример диаграммы классов

Другие примеры диаграммы классов представлены в источниках [3] и [4].

В отчете необходимо представить диаграмму классов, отражающую используемый паттерн проектирования, и краткое описание задачи, которую решает данный паттерн.

Как было указано выше, диаграмма классов является отправной точкой для построения некоторых других UML диаграмм. Для начала рассмотрим диаграмму компонентов (рисунок 2).

> **Диаграмма компонентов (component diagram)** демонстрирует инкапсулированные классы и их интерфейсы, порты и внутренние структуры, состоящие из вложенных компонентов и коннекторов. Диаграммы компонентов описывают статическое представление дизайна системы. Они важны при построении больших систем из мелких частей (UML отличает диаграмму составной структуры (composite structure diagram), применимую к любому классу, от компонентной диаграммы).

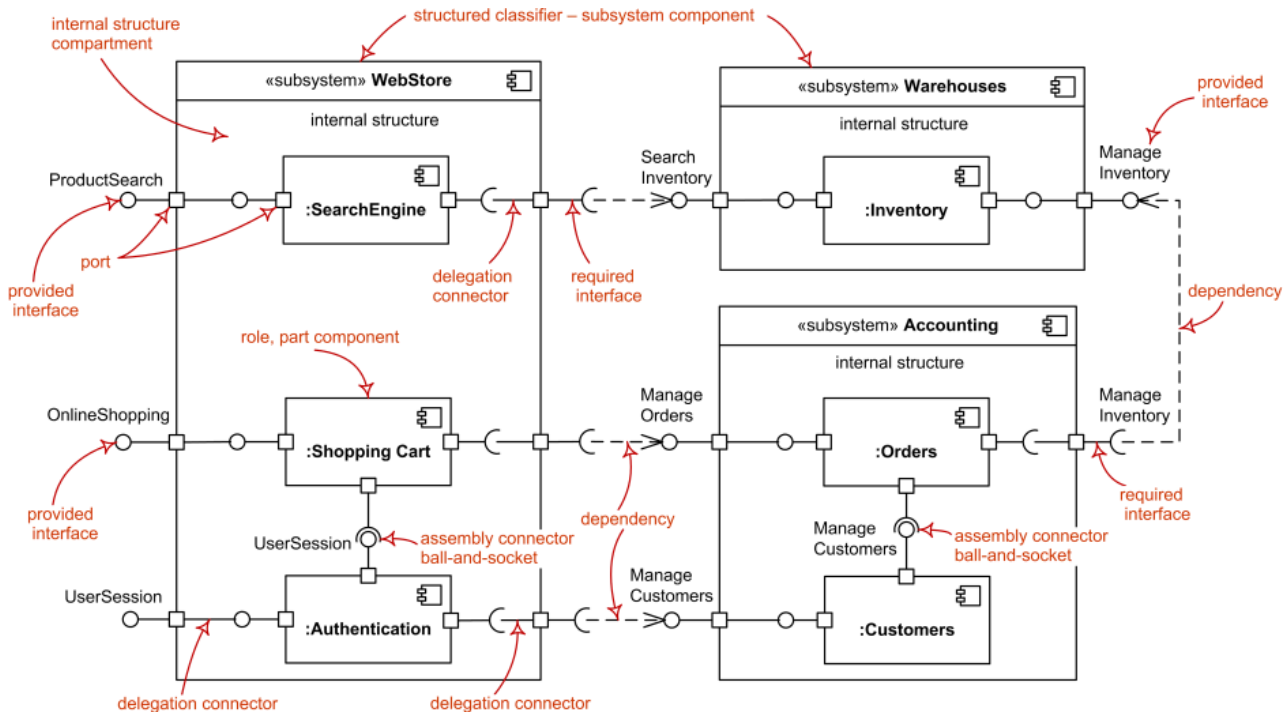

Рисунок 2 – Пример диаграммы компонентов

Одним из основных элементов данной диаграммы является компонент. **Компоненты** позволяют инкапсулировать части системы, чтобы уменьшить количество зависимостей, сделать их явными, а также повысить взаимозаменяемость и гибкость на случай, если система должна будет изменяться в будущем. Хороший компонент наделен следующими характеристиками:

- инкапсулирует сервис с хорошо очерченным интерфейсом и границами;
- имеет внутреннюю структуру, которая допускает возможность ее описания;
- не комбинирует несвязанной функциональности в пределах одной единицы;
- организует внешнее поведение, используя интерфейсы и порты в небольшом количестве;
- взаимодействует только через объявленные порты.

Более подробное описание других элементов диаграммы компонентов, а также примеры построения представлены в источнике [2, с.206-220].

В отчете необходимо представить диаграмму компонентов и краткое описание ее элементов.

Диаграмма размещения (deployment diagram) показывает конфигурацию узлов-процессоров, а также размещаемые на них компоненты. Диаграммы размещения дают статическое представление размещения архитектуры. Узлы, как правило, содержат один или несколько артефактов.

Диаграммы размещения (рисунок 3) – это один из двух видов диаграмм, используемых при моделировании физических аспектов объектно-ориентирован-

ной системы. Такая диаграмма представляет конфигурацию узлов, где производится обработка информации, и показывает, какие артефакты размещены на каждом узле.

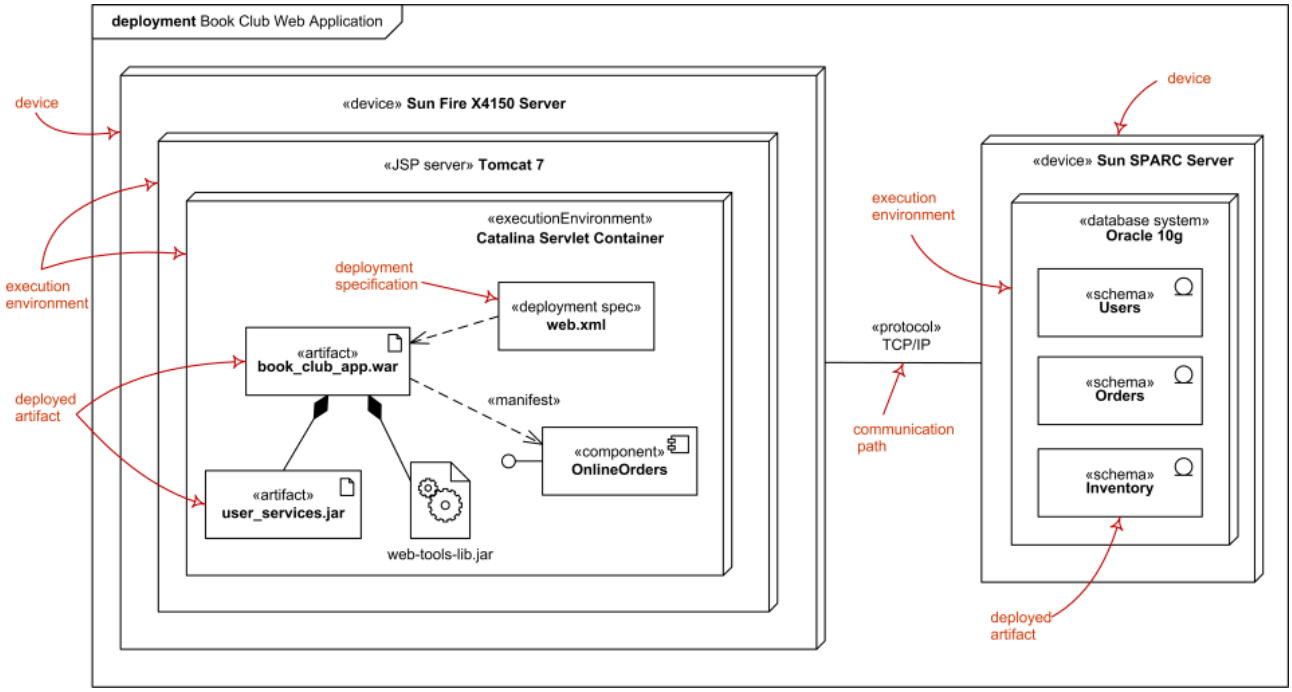

Рисунок 3 – Пример диаграммы размещения

Диаграммы размещения используются для моделирования статического представления системы с точки зрения размещения. В основном под этим понимается моделирование топологии аппаратных средств, на которых работает система. По существу, диаграммы размещения – это просто диаграммы классов, сосредоточенные на системных узлах.

Диаграммы размещения важны не только для визуализации, специфицирования и документирования встроенных, клиент-серверных и распределенных систем, но и для управления исполняемыми системами с использованием прямого и обратного проектирования.

Диаграмму размещения можно проигнорировать только в том случае, если разрабатываемое программное средство будет исполняться исключительно на одной машине, обращаясь при этом лишь к стандартным устройствам на этой же машине, управление которыми полностью возложено на ОС.

Подробнее с диаграммами размещения можно ознакомиться в источнике [2, с.379-386, с.427-438].

В отчете необходимо представить диаграмму размещения и ее краткое описание.

В случае, если разрабатываемое программное средство будет иметь большое количество классов, интерфейсов, узлов, компонентов и других элементов, целесообразно построение диаграммы пакетов (рисунок 4).

Диаграмма пакетов (package diagram) показывает декомпозицию самой модели на организационные единицы и их зависимости.

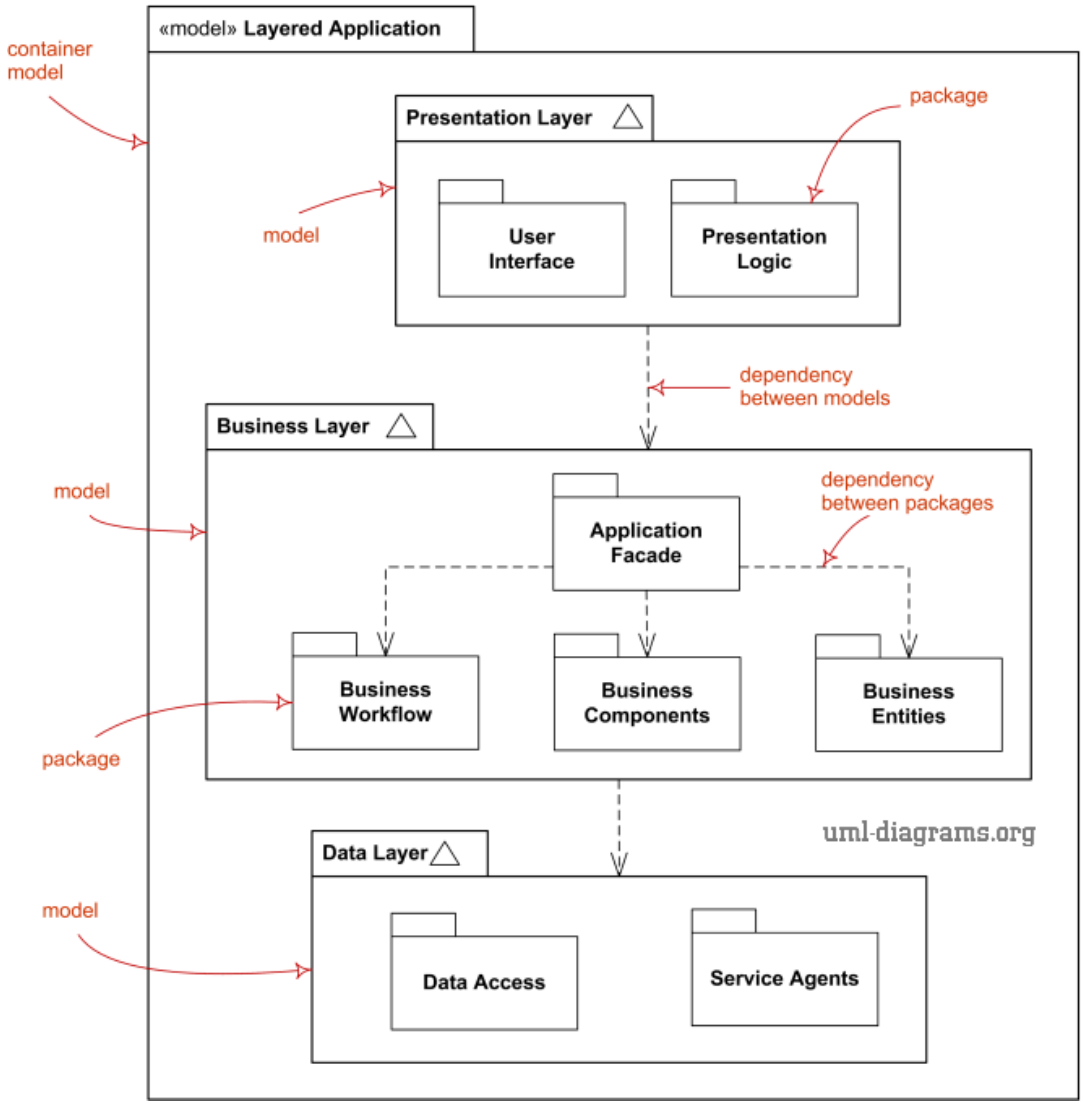

Рисунок 4 – Пример диаграммы пакетов

Основным субъектом данной диаграммы является пакет – способ организации элементов модели в блоки, которыми можно распоряжаться как единым целым. Хорошо структурированный пакет объединяет семантически близкие элементы, которые имеют тенденцию изменяться совместно. Такие пакеты характеризуются слабой связанностью и высокой согласованностью, причем доступ к содержимому пакета строго контролируется. Объединение частей модели в блоки упрощает процесс масштабирования системы в будущем.

Более детальное представление диаграмм компонентов приведено в источнике [2, с.178-189].

Проектирование пользовательского интерфейса

Проектирование пользовательского интерфейса – это создание тестовой версии приложения. Это начальный этап разработки пользовательского интерфейса, когда выполняется распределение функций приложения по экранам,

определяется макета экрана, содержимое, элементы управления и их поведение. Как результат, получается динамичный прототип интерфейса, который можно использовать для тестирования юзабилити или начала разработки приложения.

В рамках лабораторной работы проектирование пользовательского интерфейса подразумевает создание планов расположения элементов на странице (англ. *wireframe*) (рисунки 5,6).

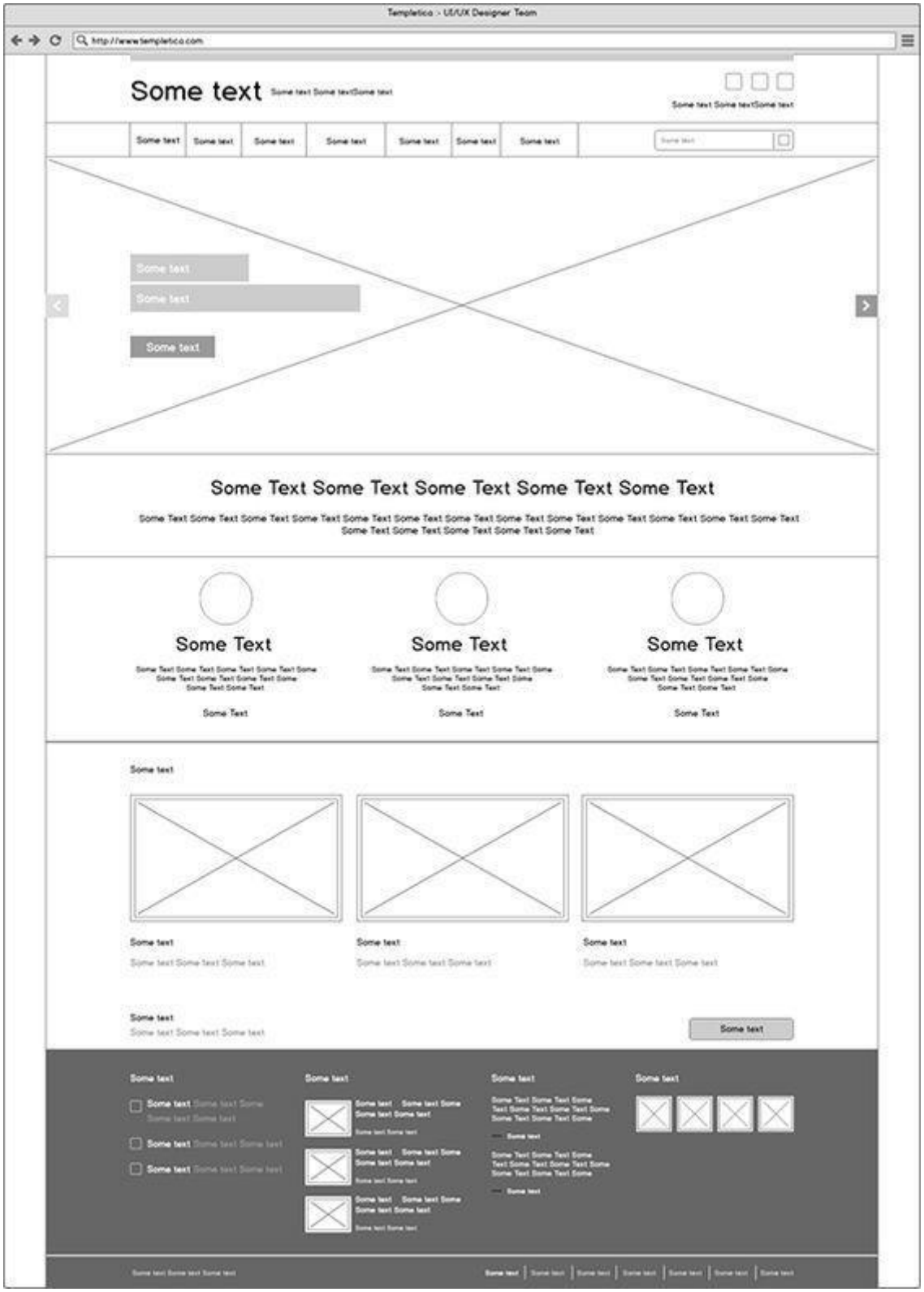

Рисунок 5 – Пример плана расположения элементов для веб-приложения

Рисунок 6 – Пример плана расположения элементов
для мобильного приложения

Принципы и примеры построения планов расположения элементов на страницах приведены в источниках [6], [7] и [8]. Для проектирования планов можно использовать, например, следующее ПО:

- Figma (*https://www.figma.com/*);
- Whimsical (*https://whimsical.com/*);
- inVision (https://www.invisionapp.com/home);
- Adobe XD (https://www.adobe.com/ru/products/xd.html);
- Zeplin (https://zeplin.io/).