В прошлом видео мы разобрали пример концепции приложения интернета вещей

То, что вИдение системы, которые мы предлагаем, согласовано с заказчиком и его устраивает.

Ее можно начать изготавливать, и мы может привлечь для этого программистов, электронщиков, начать закупать комплектующие и т.п.

Но им же всем надо сказать, что программировать, какая электроника нужна, какие комплектующие закупать, и что с этим все потом делать, чтобы собрать в единую систему, которая бы правильно работала…

Другими словами, нам нужен проект.

Как же нам перейти к проектированию?

Как уже не раз говорили, основой для начала проектирования является архитектура.

Архитектура - это про «что там внутри», Ну, в принципе

Если говорить в терминах системной инженерии, то определение системы (system definition),
исходя из требований к ней,
описывает, как система будет вести себя в окружающем мире, тогда как архитектура дает представления о принципах ее устройства.

И именно архитектура становится основой для последующего проекта.

Определение архитектуры – это принятие решений о концепциях, которые лягут в ее основу.

Это самые важные решения в том смысле, что если возникнет необходимость впоследствии какие-то из этих решений изменить, то придется переделывать всю систему.

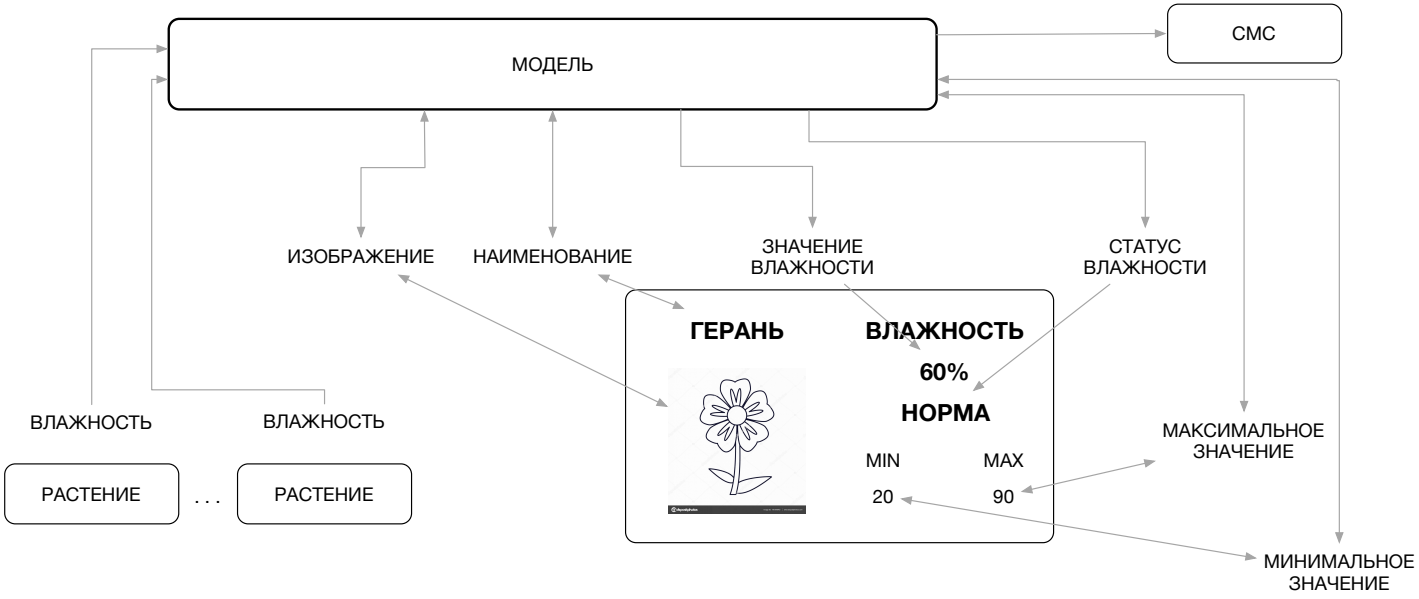
Вслед за Вигерсом повторим, что архитектура — это основные решения по структуре системы, включающая программные средства и оборудование, взаимосвязи между этими компонентами и поведение компонентов, видимое другим компонентам.

Итак, перейдем к разработке архитектуры системы их нашего примера.

Но прежде договоримся, а что будет критерием того, что архитектура выстроена правильно?

Мы об этом говорили: решения в отношении архитектуры можно считать принятыми, если понятно, как поделить работу между разработчиками подсистем, которые начнут, в свою очередь, прорабатывать их архитектуру, чтобы сформировать задания для разработчиков входящих в них элементов и так далее.

Итак, в прошлой части мы остановились на следующей структуре приложения и теперь нам надо прикинуть архитектуру модели данных.



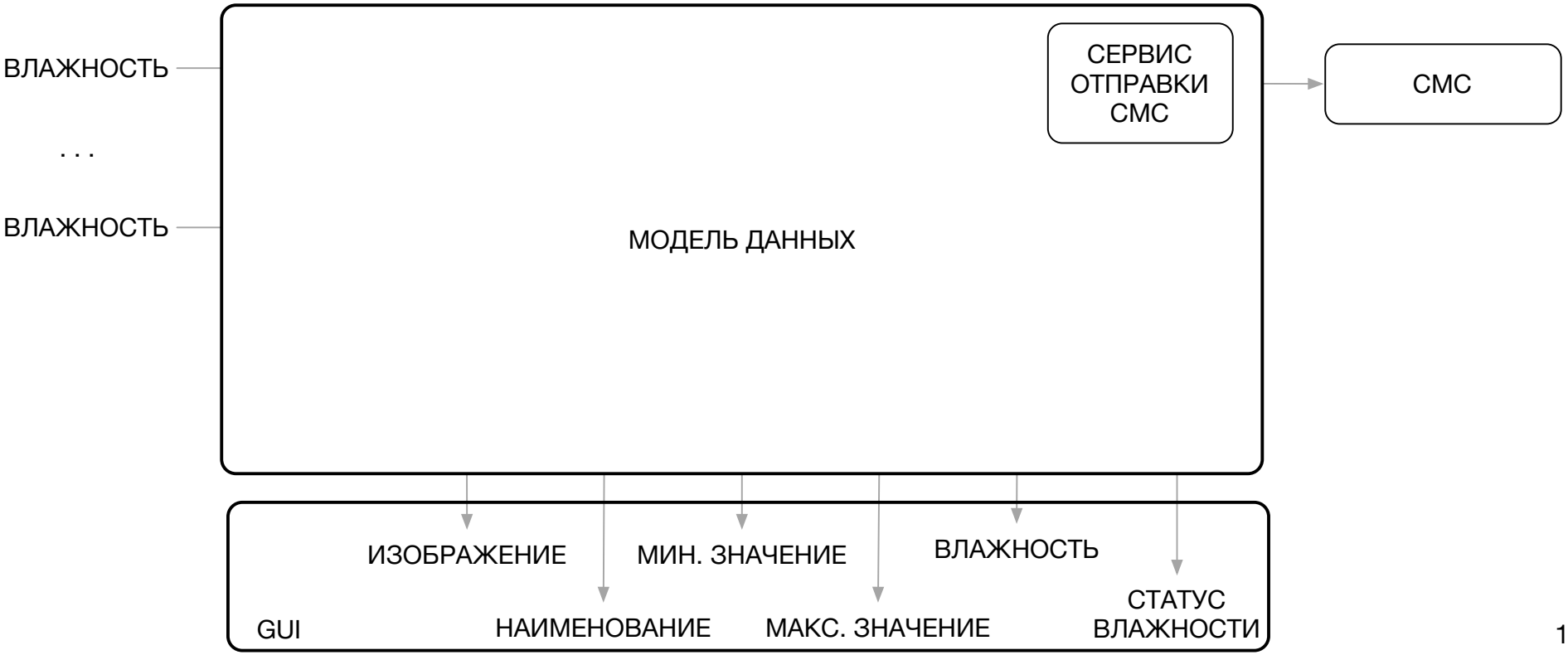
Здесь у нас модель пока является неким черным ящиком, который нам надо открыть и посмотреть, что там внутри.

Для удобства приведем ее немного к другому формату, оставляя все те же взаимодействия, поскольку они определены системными требованиями: вот входе у нее информация с датчиков влажности, она взаимодействует с пользовательским графическим интерфейсом и может отправлять СМС.

Начнем открывать черный ящик.

Что мы знаем про то, что у него внутри?

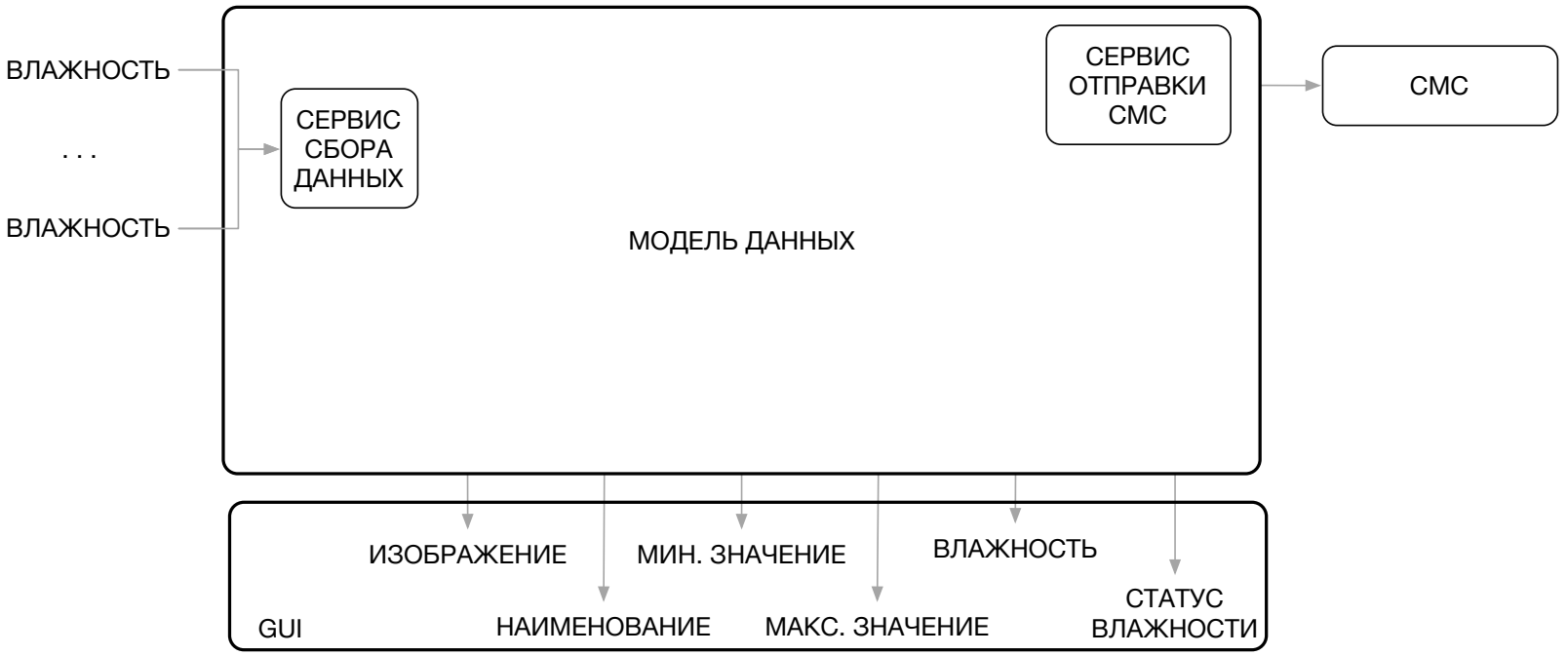
Самое очевидное: если у нас должно отправляться смс, то должны быть какая-то служба, которая его отправляет:



Что еще очевидно?

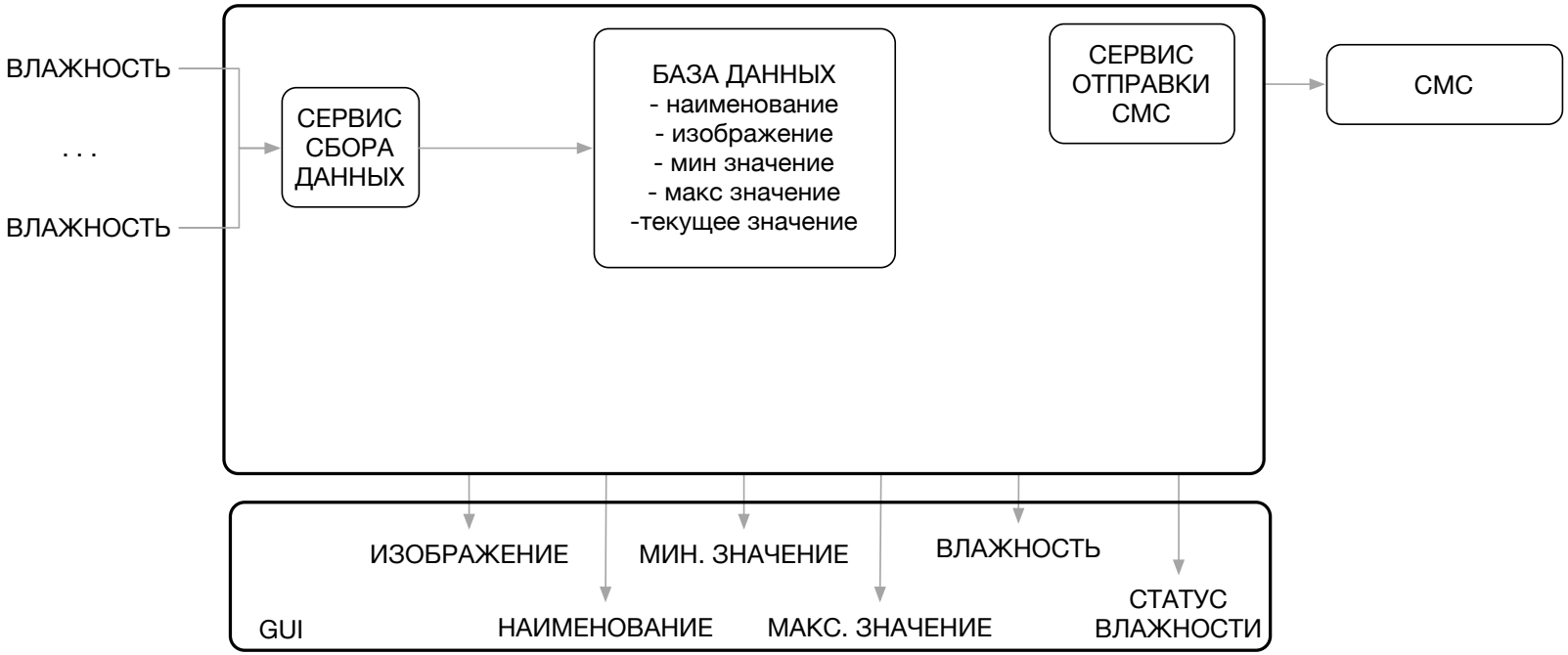
Нам надо выбрать стратегию представления данных, и датчики у нас такие, что каждый отправляет свою информацию в приложении независимо.

Сделаем один общий сервис для сбора данных и используем стратегию МНОГИЕ-К-ОДНОМУ.



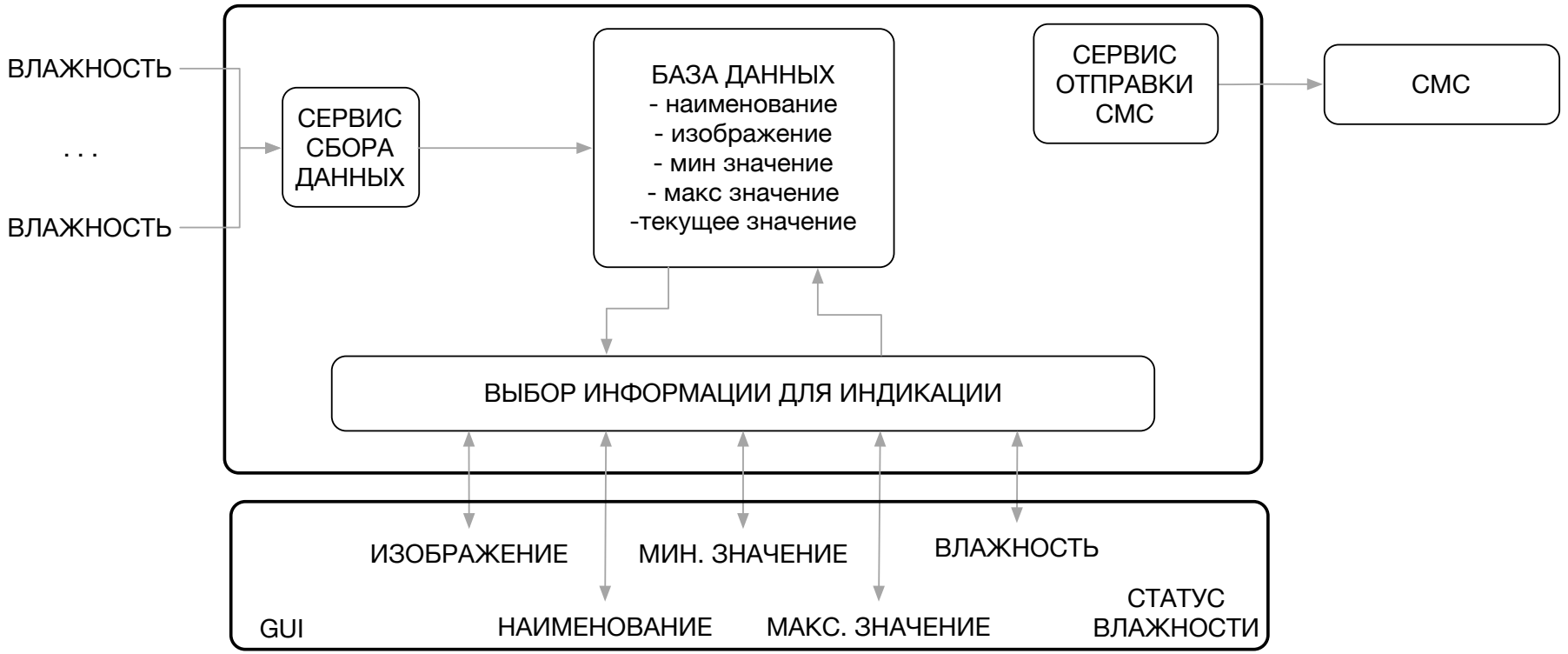
Для хранения текущей информации логично организовать базу данных, которая будет содержать

- наименование
- изображение
- мин значение
- макс значение
- и текущее текущее значение влажности



Идем дальше… Графический интерфейс у нас сделан так, что из множества растений у нас на экран выводятся данные только одного.

Соответственно, нужен какой-то логической блок, который бы выдавал в интерфейс нужные данные, получая их из базы.

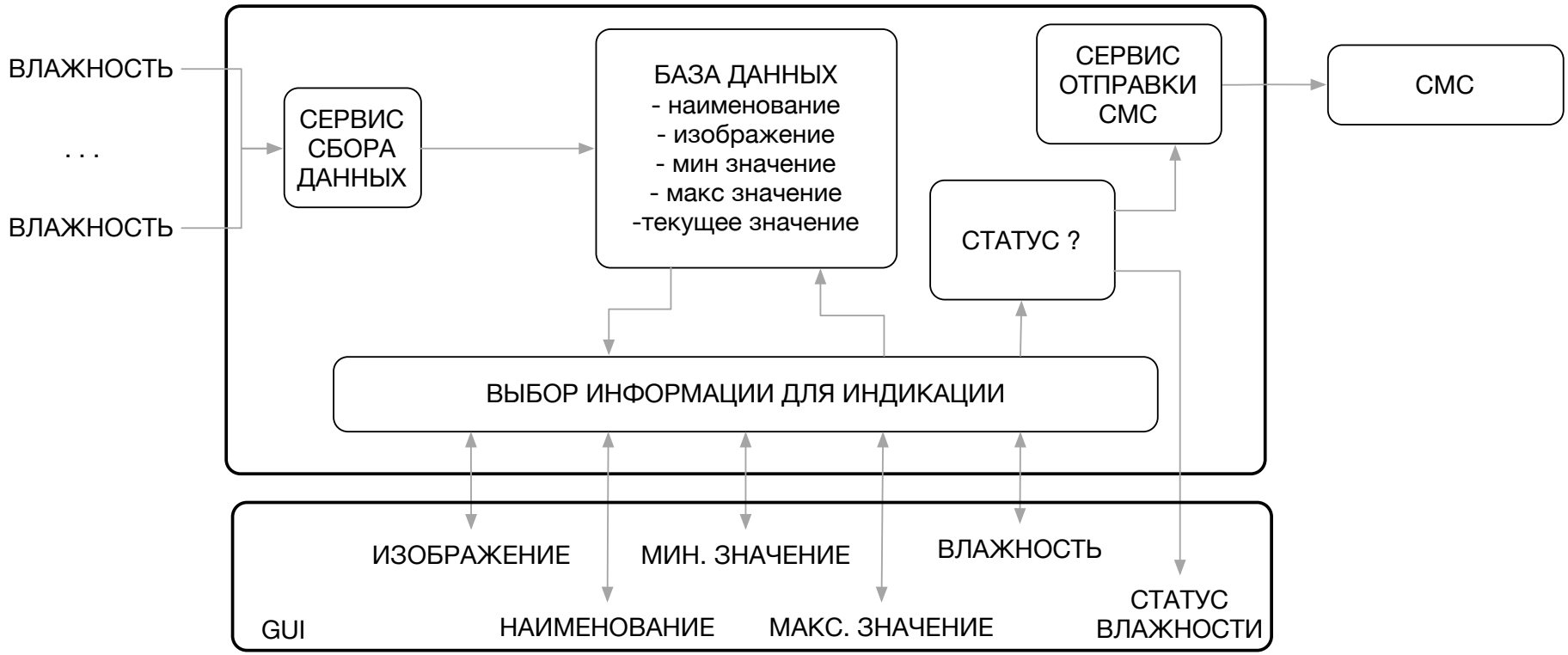


В таком виде приложение будет выдавать нам текущее значение влажности почвы для выбранного растения. Отлично!

Какое еще требование не удовлетворено? Выдача предупреждений!

Значит нам не хватает в структуре элемента, который бы сравнивал текущее значение важность с минимальным и максимальным, определял бы, не сухо ли, и не слишком ли влажно, и если так - выводил бы эту информацию на графический интерфейс и отправлял бы смс.

Добавим такой блок! Вот, что получаем в результате



В принципе, все достаточно просто…

Архитектура, в отличие от концепции, описывается уже гораздо более строго и, как правило, сопровождается разного рода схемами, диаграммами и пр.

Какая же практическая польза от такой «картинки»?

Среди прочего - исключительно прикладная: правильно выстроенная архитектура автоматически дает вам перечень работ, необходимых для создания системы!

Посмотрите, каждый элемент, каждый блок на схеме может быть независимо отдан в разработку отдельному исполнителю - будь то специалист в офисе или на удаленке, или отдельной команде или подразделению, либо даже стороннему разработчику на подряде.

И важно при этом, что эти работы могут выполняться совершенно независимо, что система может собираться по частям, и именно правильно выстроенная архитектура системы делает возможным применение в разработке такого рода систем гибких методологий - agile, scrum, Kanban, о чем мы еще подговорим отдельно.

Смотрите, мы можем поручить кому-то написать сервис сбора данных, а кому-то - сервис отправки смс.

Кто-то может параллельно делать базу данных, а кто-то - выстраивать логику работы с данными и «поведение» системы.

И, естественно, кто-то - дизайнеры - в это же время, параллельно и независимо - будет делать все интерфейсы. А кто-то займется датчиками и их подключением.

Причем весь вывод данных можно будет оттестировать как будет готов блок вывода, и совсем не надо ждать реальных данных.

Для проверки работы блока определения статуса - не надо ждать подключения реальных устройства и так далее.

Такая схема дает нам взаимосвязи между этими компонентами, но чтобы разработку можно было организовать, как описано, у нас должны быть правильно определены все компоненты, и поведение компонентов, видимое другим компонентам.

И тут мы снова возвращаемся к формату описания, который обсуждали, когда говорили о требованиях.

Нам потребуются спецификации, где по каждому объекту и по каждой связи будет расписано

- Наименование объекта (Thing), описание и другая необходимая информацию.
- Перечень его свойств (Properties).
- Перечень событий (Events) или предупреждений (Alerts), которые инициирует объект.

- Перечень событий или предупреждений, на которые он подписан и процедуры соответствующих подписок.
- Перечень и процедуры необходимых сервисов.
- Описание взаимодействия с пользователем и предложения по дизайну веб-страницы приложения.

- Описание параметров подключения к приложению внешних физических/виртуальных объектов.
- Структурные схемы подключения вешних объектов и другая информация, необходимая для их подключения к приложению.

Причем работа над архитектурой является итеративной. К примеру, если при описании процедур выясняется, что это требует наличия у объекта свойств, не описанных ранее, эти свойства добавляются в общее описание системы, идет уточнение компонентов и их взаимосвязей и так далее.

Name: **Shipyard** Type: Thing

Property Name	Base Type	Unit	History?	Persist?
PowerProduced	Number	kW	Yes	Yes
PowerConsumed	Number	kW	Yes	Yes
PowerStored	Number	kWh	Yes	No
If History is Yes, Value Stream Name:	VS_Username	**When is data logged?** When PowerStored value changes by 100; PowerProduced and PowerConsumed value changes by 0		

Which Roles Need Run Time Permissions?	**Property Read/Write**	**Why?**
Power.General	Read	To see the amount of power stored
ShipyardAdminGrp	Read	To see the amount of power stored

When will data be sent/retrieved?

Every 10 seconds

Service Name	Purpose
Rollup	Goes out and fetches the aggregates of the PowerProduced, PowerConsumed and PowerStored property values across the entire Power System.
Event Name	**Purpose**
Timer	Since the Shipyard inherits from a timer based thing template, we can fire a timer event every ten seconds.
Subscription	**Purpose**
Timer	The subscription to the timer event will fire the Rollup service, gathering the data every ten seconds from the power producers, power consumers and the power storage thing.

Если у вас правильно выстроена архитектура и в необходимом объеме определены требования к компонентом и их взаимодействию, значит вы можете «раскидать» работы по исполнителям в небольшом проектам, либо передать в проектирование подсистемы в проекте большом.

А вот одним из способов проверки правильности построения архитектуры является то, насколько перечень работ «бьется» с перечнем элементов, заявленных в архитектуре.

Важно, что мы тут говорим об общем подходе и максимально «на пальцах» и конкретный способы разделения системы на части - это отдельная тема.

Более того, разные заинтересованные лица - стейкхолдеры - по-разному выделяют части в системе и могут иметь дело с разными их представлениями - компоненты, модули, размещения, процессы, элементы, связи и пр. - на эту тему есть огромное количество специальной литературы и учебных материалов.

Сейчас мы просто показали общий подход, причем для небольших проектов им можно пользоваться непосредственно, а если к вам в работу попадает крупный проект - наверняка вы привлечете специалистов, но, по крайне мере, уже будете понимать, о чем они…

Естественно, все рекомендации и подходы, о которых мы говорили в частях курса, посвященных сбору требований и их документированию, тут актуальны.

И именно архитектура проекта, передаваемая исполнителю для реализации, со всеми необходимыми требованиями, описанием из функционирования и способов приемки, и является основой «классического» технического задания, в том числе по ГОСТу, о чем мы говорили ранее.

А вот как организовать работу над проектом, используя гибкие методологии разработки, мы поговорим в следующем видео.