**Уточнение требований к системе**

Разработка программного обеспечения начинается с анализа требований к функциональности, которые указаны в техническом задании. В соответствии с техническим заданием у разрабатываемой системы должно быть три типа пользователей – любой аутентифицированный пользователь, пользователь с правами модератора и пользователь с правами администратора. Аутентифицированный пользователь должен иметь возможность создавать заявки на добавление ПО, просматривать свои заявки на добавление ПО, просматривать каталог файлов и выгружать с сервера файлы. Модератор должен

Для уточнения требований к функциональности системы необходимо рассмотреть варианты ее использования. Вариант использования представляет собой характерную процедуру применения разрабатываемой системы конкретным действующим лицом [Иванова]. Выявленные варианты использования разрабатываемой системы представлены на диаграмме вариантов использования (рисунок 21).

Рисунок 21 – Диаграмма вариантов использования разрабатываемой системы.

**Выбор способа проектирования**

Существует два способа проектирования программных систем:

* Структурное проектирование;
* Объектно-ориентированное проектирование.

Структурное проектирование основано на алгоритмической декомпозиции, то есть разбиении системы на элементарные функции. Система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее. Процесс разбиения продолжается вплоть до конкретных процедур. Каждый модуль системы выполняет один из этапов общего процесса. При этом создаваемая система сохраняет целостное представление, все составляющие компоненты в ней взаимосвязаны.

Объектно-ориентированное проектирование основано на объектной декомпозиции. При такой декомпозиции каждый объект обладает своим собственным поведением и каждый из них моделирует некоторый объект реального мира. Программа, полученная при использовании такого подхода, описывает взаимодействие объектов.

Эти способы противоположны друг другу, поэтому нельзя сконструировать сложную систему использую одновременно оба способа. Объектная декомпозиция имеет ряд преимуществ над алгоритмической декомпозицией:

* Объектная декомпозиция уменьшает размер программных систем за счет повторного использования общих механизмов, что приводит к существенной экономии выразительных средств.
* Объектно-ориентированные системы более гибки и проще эволюционируют со временем, потому что их схемы базируется на устойчивых промежуточных формах. Действительно, объектная декомпозиция существенно снижает риск при создании сложной программной системы, так как она развивается из меньших систем, в которых мы уже уверены.
* Объектная декомпозиция помогает нам разобраться в сложной программной системе, предлагая нам разумные решения относительно выбора подпространства большого пространства состояний. [Груди Буч]

На основе приведенных преимуществ был выбран объектно-ориентированный подход. При выполнении объектной декомпозиции в результате анализа предметной области были выделены следующие сущности: пользователь, файл, версия файла, категория, свойство, свойство файла, свойство версии файла, категория файла, заявка и файл заявки. Для описания этих сущностей будут использованы классы.

**Разработка базы данных**

В соответствии с техническим заданием, вся информация в системе должна храниться в СУБД PostgreSQL. В результате объектной декомпозиции в разрабатываемой системе были выделены сущности, каждая из которых будет отображена соответствующей реляционной таблицей.

Таблица user необходима для хранения информации о пользователях в системе. В ней будут храниться ФИО, телефон, электронный адрес, номер и название отдела и адрес. Таблица file предназначена для хранения общей информации о файлах и для объединения версий файлов. Для хранения версий файлов предназначена таблица fileVersion, в которой храниться информация по конкретной версии файла: размер, номер версии, хэш-сумма, дата загрузки. Таблица category хранит информацию о категориях, которые предназначены для группировки файлов. Для того чтобы категории можно было выстраивать в виде дерева в таблице присутствует поле parent, которое указывает на id родительской категории. Так же, в таблице храниться название категории и ее порядковый номер. Для хранения связей файлов с категориями предназначена таблица fileCategory. В соответствии с техническим заданием, у файлов должны быть свойства. Для их хранения служит таблица property. Чтобы свойства можно было привязывать как к файлу, так и версии файла, необходимы две таблицы fileProperty и fileVersionProperty. Для сохранения заявок на добавление ПО предназначена таблица request. Так как к одной заявке может быт прикреплено сразу несколько файлов, то файлы заявки были вынесены в отдельную таблицу requestFile. Таблица log предназначена для ведения журнала приложения.

Полученная в результате проектирования логическая схема базы данных представлена на рисунке 22.

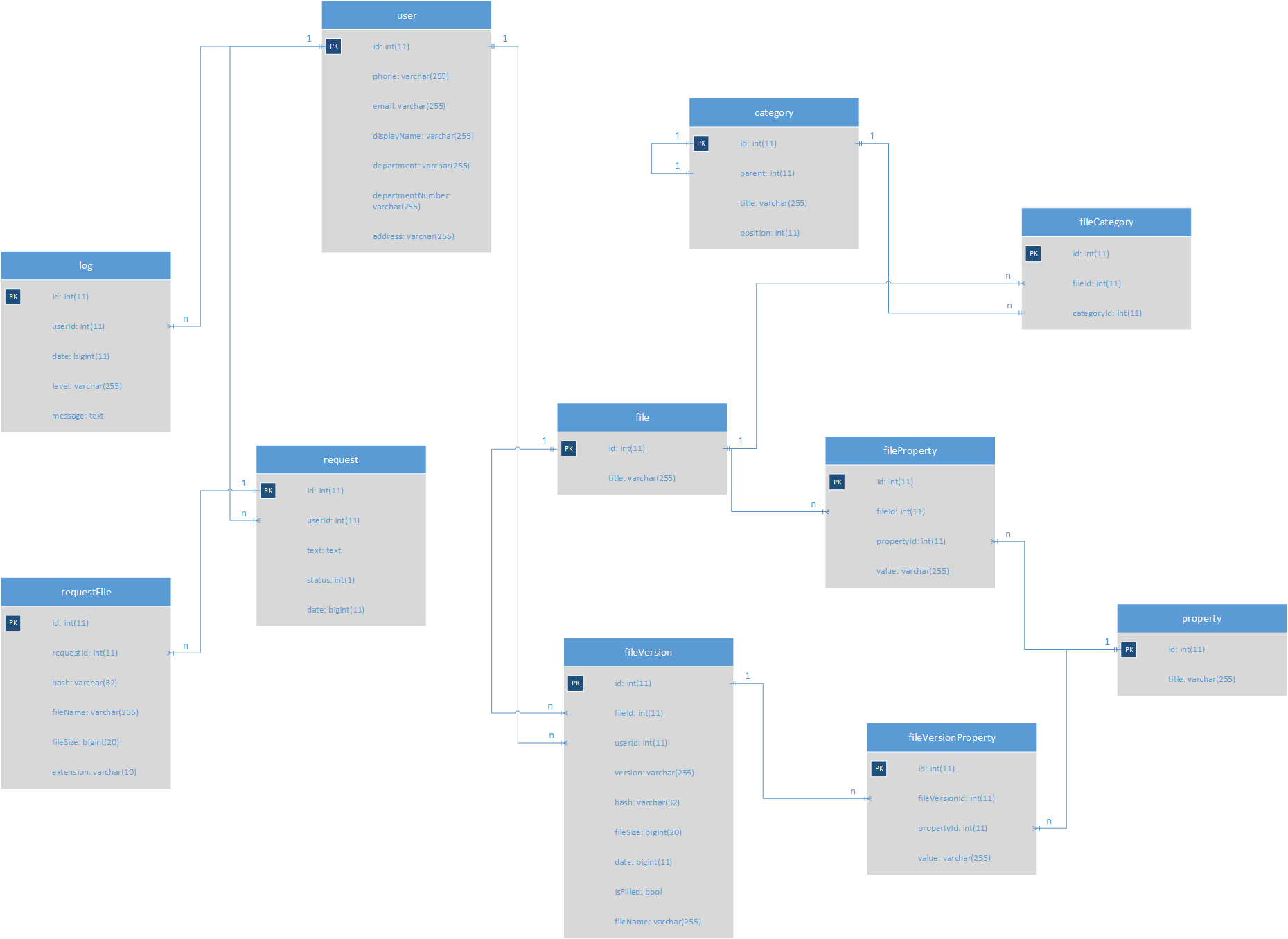


Рисунок 22 – Логическая схема базы данных.

**Разработка классов**

Для каждой сущности, выделенной в результате объектной декомпозиции, необходимо разработать классы. Разрабатываемые классы будут содержать работу с базой данных: чтение записей из базы данных, добавление и обновление записей в базу данных и удаление записей из базы данных. Получается, что в классах будут однотипные методы, но у них будет разная реализация. Поэтому, целесообразно вынести такие методы в базовый класс ModelInterface, который будет является интерфейсом. В нем содержаться методы:

* update() для обновления записи в базе данных;
* add() для сохранения новой записи в базу данных;
* delete() для удаления записи из базы данных;
* validate() для проверки правильности заполнения полей.

Структура интерфейса ModelInterface представлена на рисунке 21. Все разрабатываемые классы выделенных сущьностей будут реализовывать интерфейс ModelInterface.

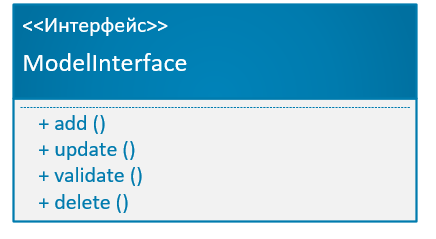


Рисунок 21 – Диаграмма класса ModelInterface.