2.3 Разработка шаблона кода

После завершения этапа проектирования объектной модели, включающего определение сущностей и установление связей между ними, была построена схема данных, отражающая архитектуру будущей информационной системы. Подход BCE (Boundary-Control-Entity – граница-управление-сущность) представляет собой подход к объектному моделированию, основанный на трехфакторном представлении классов. В правильно спроектированной иерархии пакетов актер может взаимодействовать только с пограничными объектами из пакета BoundaryPackage, объекты-сущности из пакета EntityPackage могут взаимодействовать только с управляющими объектами из ControlPackage и управляющие объекты из ControlPackage могут взаимодействовать с объектами любого типа. Основным преимуществом подхода BCE является группирование классов в виде иерархических уровней. Это способствует лучшему пониманию модели и уменьшает ее сложность [8].

Пакет Entity содержит классы, отвечающие за представление данных, которые используются в системе. В него входят сущности PotentialClients, RecordedClients, Documents, StatusClient, ClientsHistory, Worker, CategoryClients, представляющие данные о лидах, зарегистрированных заказчиках, ответственных рабочих, статусов и документах, а также историю изменений. Эти элементы являются основой хранения информации и используются для выполнения операций бизнес-логики.

Пакет Boundary включает компоненты, которые обеспечивают внешнее взаимодействие пользователя с системой. К ним относятся ClientPage, отображающая информацию о лидах, ClientForm, предназначенная для добавления новых лидов, формы для загрузки документов и страницы просмотра детальной информации, включая историю изменений. Эти элементы реализуют интерфейсные функции и служат точками входа для взаимодействия человека с программным обеспечением.

Пакет Control содержит классы, реализующие бизнес-логику и координирующие связь между пользовательскими компонентами и сущностями данных. Среди них находятся RecordController, отвечающий за обработку форм и сохранение данных, DocumentController, управляющий документами, а также Manager, выполняющий отправку уведомлений менеджерам при создании нового лида или изменении его состояния. Контроллеры являются связующим звеном между уровнем представления и уровнем данных, обеспечивая выполнение логики приложения.

После декомпозиции всех элементов была сформирована итоговая диаграмма компонентов, отражающая структуру системы управления лидами и

документами заказчиков и демонстрирующая взаимодействие boundary-компонентов, контроллеров и сущностей в едином архитектурном пространстве. Эта интегрированная схема позволяет получить целостное представление о функционировании системы и служит основой для дальнейшей автоматической генерации кода и последующей программной реализации. Она представлена на рисунке 2.7.

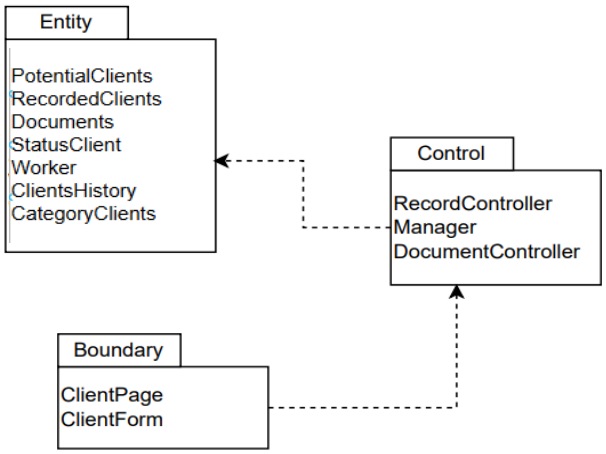

Рисунок 2.7 – Схема данных пакетов

Ниже представлен код C++ для Entity-класса Worker.

```
#pragma once
#include <string>
class Worker {
private:
    int id;
    std::string name;
    std::string phone;
public:
    Worker() : id(0) {}
    Worker(int id, const std::string& name, const std::string& phone)
        : id(id), name(name), email(email) {}
    int getId() const { return id; }
    std::string getName() const { return name; }
    std::string getPhone() const { return phone; }
    void setName(const std::string& n) { name = n; }
    void setPhone(const std::string& e) { phone = e; }
};
```