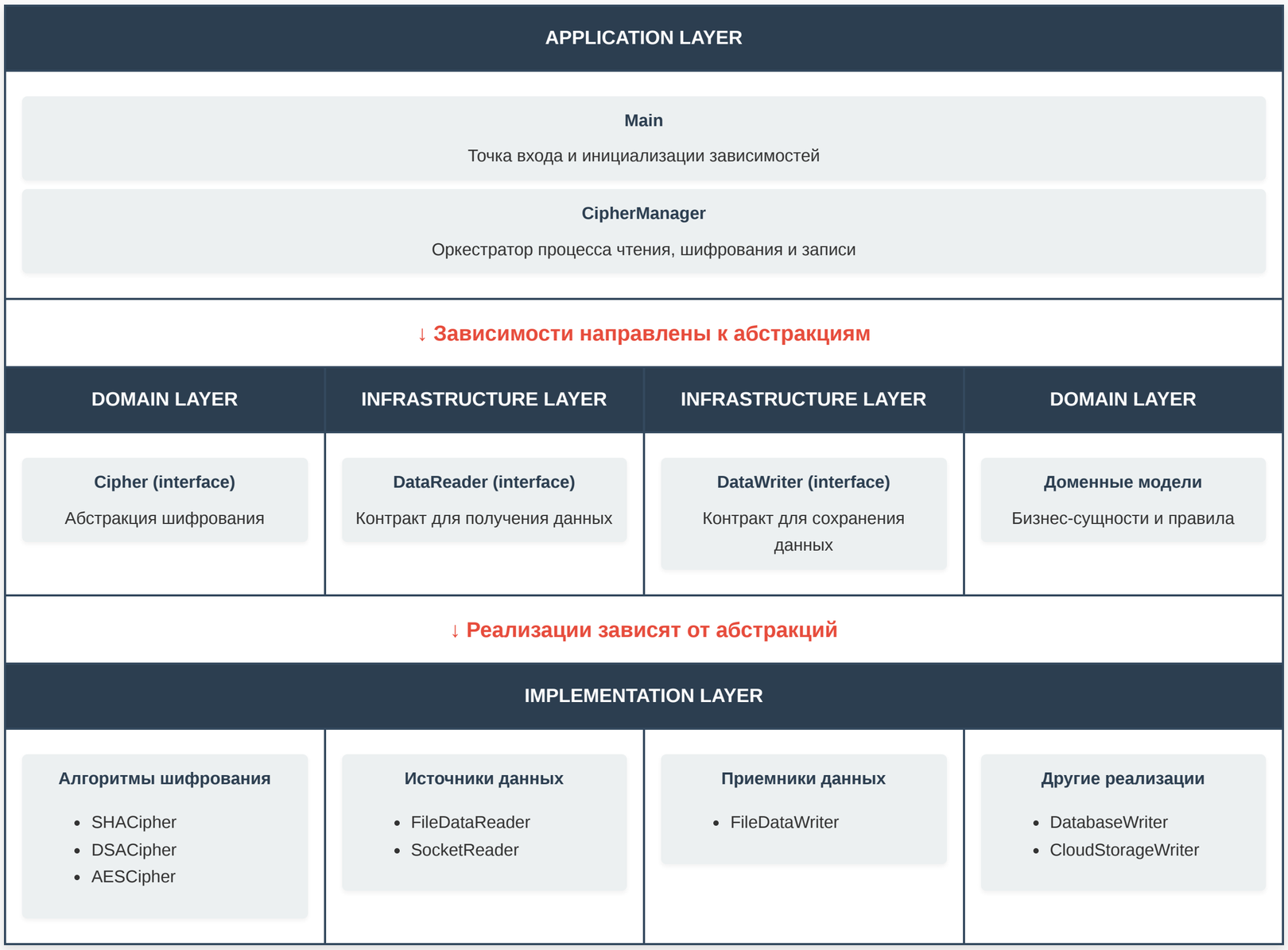
Архитектура системы шифрования данных

Диаграмма компонентов



Принципы компоновки

Common Closure Principle (CCP)

Классы, которые меняются по одинаковым причинам и в одинаковое время, группируются вместе.

- Все алгоритмы шифрования сгруппированы
- Все реализации IO операций сгруппированы отдельно

Dependency Inversion Principle (DIP)

Зависимости направлены к абстракциям, а не к реализациям.

- Высокоуровневые модули не зависят от низкоуровневых
- Абстракции не зависят от деталей

Stable Dependencies Principle (SDP)

Зависимости направлены в сторону более устойчивых компонентов.

- Domain Layer - наиболее устойчивый компонент
- Implementation Layer - наиболее изменчивый компонент

Low Coupling & High Cohesion

Минимальные зависимости между компонентами, сильная связанность внутри компонентов.

- Компоненты слабо связаны между собой
- Классы внутри компонентов тесно связаны

Преимущества архитектуры

- **Легкость тестирования** - Компоненты можно тестировать изолированно, используя моки и стабы для зависимостей
- **Гибкость** - Легко добавлять новые алгоритмы шифрования и источники данных без изменения существующего кода
- **Поддержка** - Изменения локализованы в соответствующих компонентах, что упрощает сопровождение
- **Масштабируемость** - Четкие границы компонентов позволяют легко расширять систему
- **Переиспользование** - Компоненты могут использоваться в других проектах благодаря слабой связанности
- **Простота понимания** - Четкое разделение ответственности делает архитектуру интуитивно понятной

Направления зависимостей

В архитектуре соблюдается такое направление зависимостей:

- **Application Layer** → **Domain Layer** - Приложение зависит от бизнес-логики
- **Application Layer** → **Infrastructure Layer** - Приложение зависит от абстракций инфраструктуры
- **Implementation Layer** → **Domain Layer** - Реализации зависят от абстракций бизнес-логики
- **Implementation Layer** → **Infrastructure Layer** - Реализации зависят от абстракций инфраструктуры