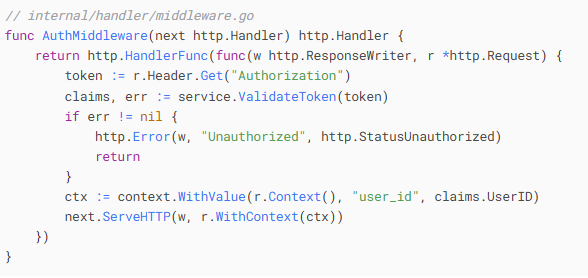
Простой http-сервер, который отображает список скважин из базы postgres, с авторизацией пользователей, есть возможность редактировать, удалять и добавлять записи. Также при редактировании и добавлении записи выполняется определенная математическая логика по обработке данных скважины и показываются результаты. Будет простой экспорт данных и отчеты.

Схема:

1. **Аутентификация (JWT)**

Пример middleware:



#### 2. ****CRUD для скважин****

Пример обработчика:



Пример использования транзакции для атомарного обновления:

func (r \*WellRepo) UpdateWithTransaction(ctx context.Context, well \*entity.Well) error {

tx, err := r.db.Begin(ctx)

if err != nil {

return err

}

defer tx.Rollback(ctx)

if err := r.Update(ctx, well); err != nil {

return err

}

*// Дополнительные операции...*

return tx.Commit(ctx)

}

Пример использования сервиса:

*// Пример в обработчике HTTP*

func (h \*WellHandler) UpdateWell(w http.ResponseWriter, r \*http.Request) {

var well entity.Well

if err := json.NewDecoder(r.Body).Decode(&well); err != nil {

http.Error(w, err.Error(), http.StatusBadRequest)

return

}

updatedWell, err := h.wellService.UpdateWell(r.Context(), &well)

if err != nil {

http.Error(w, err.Error(), http.StatusInternalServerError)

return

}

w.Header().Set("Content-Type", "application/json")

json.NewEncoder(w).Encode(updatedWell)

}

Для тестирования можно использовать mock репозитория, как было показано ранее.

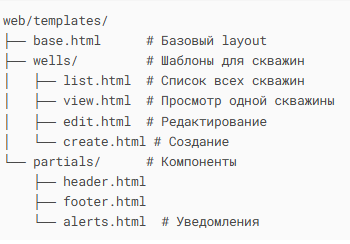


### **4. Закрепление контекста в новом чате**

Если вы обсуждаете код или проект, можно:

* В начале нового чата **явно указать контекст**:

"Продолжаем обсуждение проекта gas\_wells\_api на Go. Ранее мы реализовали репозиторий и сервис. Вот текущая структура: [вставьте код или описание]."



1. **Адаптивность**:

css

Copy

Download

@media (max-width: 768px) {

.wells-grid {

grid-template-columns: 1fr;

}

.well-form {

padding: 0 15px;

}

}

1. **Кастомные скроллбары**:

css

Copy

Download

::-webkit-scrollbar {

width: 8px;

}

::-webkit-scrollbar-track {

background: var(--light);

}

::-webkit-scrollbar-thumb {

background: var(--primary);

border-radius: 4px;

}

1. **Интерактивные элементы**:

css

Copy

Download

.btn-hover-effect {

position: relative;

overflow: hidden;

}

.btn-hover-effect:after {

content: "";

position: absolute;

top: 50%;

left: 50%;

width: 5px;

height: 5px;

background: rgba(255, 255, 255, 0.5);

opacity: 0;

border-radius: 100%;

transform: scale(1, 1) translate(-50%);

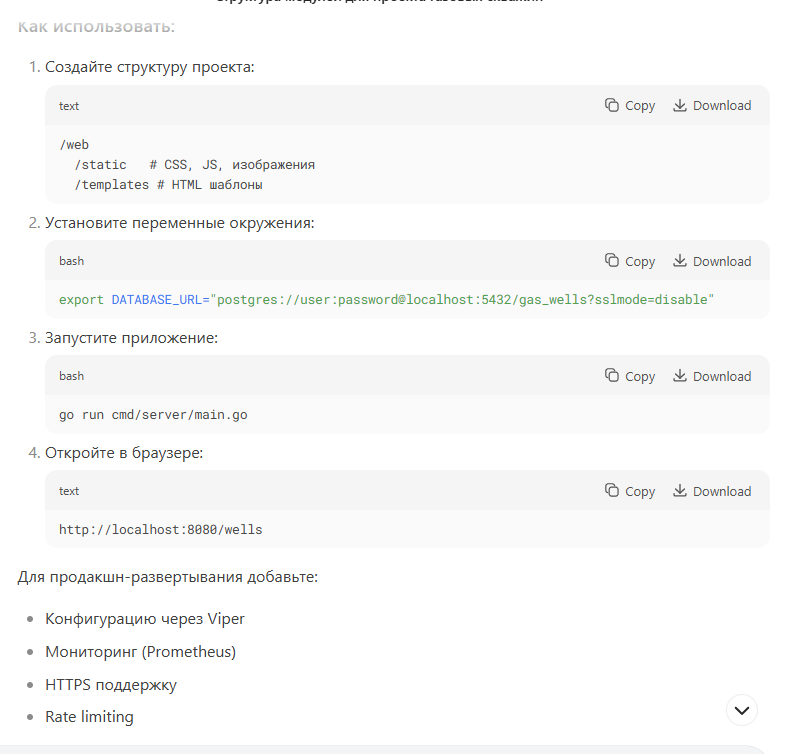
transform-origin: 50% 50%;

}

.btn-hover-effect:focus:not(:active)::after {

animation: ripple 1s ease-out;

}



### **Ключевые преимущества:**

1. **Гибкая конфигурация**:
   * Поддержка разных окружений (dev/stage/prod)
   * Типизированные значения (duration, int, bool)
   * Значения по умолчанию
2. **Мощная валидация**:
   * Готовые правила для распространенных сценариев
   * Простое добавление кастомных правил
   * Поддержка сложных условий
3. **Безопасность**:
   * Валидация перед обработкой данных
   * Защита от инъекций через строгую типизацию
4. **Масштабируемость**:
   * Легко добавлять новые параметры конфигурации
   * Простое расширение правил валидации
5. **Интеграция**:
   * Работает со стандартными библиотеками Go
   * Подходит для REST API и HTML форм

Для использования:

1. Создайте pkg/config и pkg/validation пакеты
2. Импортируйте их в нужных местах
3. Используйте .env файл для настроек окружения
4. **Следующие шаги**:

* Добавьте валидацию параметров пагинации (limit/offset)
* Реализуйте метод Count() для полного количества записей
* Обновите Swagger-документацию (если используете)

1. **Если возникнут вопросы**:

* По архитектуре - обращайтесь, помогу с организацией кода
* По работе с БД - помогу оптимизировать запросы
* По тестированию - покажу как писать тесты для репозиториев

1. **Полезно перед сном**:

* Закоммитьте текущие изменения
* Набросайте TODO-list на завтра
* Проверьте, чтобы go build проходил без ошибок