Фаза 1: Настройка Проекта Lynkr

🎯 Цели Фазы 1

- [x] Репозиторий создан: https://github.com/saidojp/Lynkr
- [ ] Инициализация Nuxt 3 проекта с TypeScript
- [ ] Настройка Supabase
- [ ] Конфигурация UI Kit (Shadcn + UnoCSS)
- [ ] Настройка инструментов разработки (ESLint, Prettier, Husky)
- [ ] Базовая дизайн-система и темы
- [ ] Структура базы данных
- [ ] Система авторизации

📋 Чек-лист Задач

✅ Шаг 1: Инициализация Nuxt 3 проекта

```bash
# Клонируем репозиторий
git clone https://github.com/saidojp/Lynkr.git
cd Lynkr

# Инициализируем Nuxt 3 проект
npx nuxi@latest init .
# или если нужно создать в текущей папке
npx nuxi@latest init --force .

# Устанавливаем зависимости
npm install
# или
pnpm install
# или
yarn install
```

✅ Шаг 2: Конфигурация TypeScript

Создаем `nuxt.config.ts`:

```typescript
```

```
// nuxt.config.ts
export default defineNuxtConfig({
  devtools: { enabled: true },

  // TypeScript конфигурация
  typescript: {
    typeCheck: true,
    strict: true
  },

  // CSS фреймворки
  css: ['@unocss/reset/tailwind.css'],

  // Модули
  modules: [
    '@nuxt/eslint',
    '@pinia/nuxt',
    '@unocss/nuxt',
    '@vueuse/nuxt',
    '@nuxtjs/supabase'
  ],

  // Настройки приложения
  app: {
    head: {
      title: 'Lynkr - Your Link Memory',
      meta: [
        { name: 'description', content: 'Minimalist link saving and organization app' },
        { name: 'viewport', content: 'width=device-width, initial-scale=1' }
      ]
    }
  },

  // Настройки рантайма
  runtimeConfig: {
    public: {
      supabaseUrl: process.env.SUPABASE_URL,
      supabaseAnonKey: process.env.SUPABASE_ANON_KEY,
    }
  }
})
```

✅ Шаг 3: Установка и настройка зависимостей

bash

```
# Основные зависимости
pnpm install @nuxtjs/supabase @pinia/nuxt @vueuse/nuxt

# UI и стилизация
pnpm install @unocss/nuxt @unocss/reset
pnpm install @nuxt/icon
pnpm install @vueuse/motion

# Инструменты разработки
pnpm install -D @nuxt/eslint eslint prettier
pnpm install -D @typescript-eslint/parser @typescript-eslint/eslint-plugin
pnpm install -D husky lint-staged

# Утилиты
pnpm install zod @vueuse/core
pnpm install lucide-vue-next
```

Важно: Если получаете ошибку с `@nuxt/eslint`, можете временно убрать его из конфигурации или использовать альтернативную настройку ESLint.

✅ Шаг 4: Настройка UnoCSS

Создаем `uno.config.ts`:

typescript

```ts
// uno.config.ts
import { defineConfig, presetUno, presetAttributify, presetWebFonts } from 'unocss'

export default defineConfig({
  presets: [
    presetUno(),
    presetAttributify(),
    presetWebFonts({
      fonts: {
        sans: 'Inter:400,500,600,700',
        mono: 'JetBrains Mono:400,500,600',
      }
    })
  ],

  theme: {
    colors: {
      // Основная палитра (ч/б стиль с акцентами)
      primary: {
        50: '#f8f9fa',
        100: '#f1f3f4',
        200: '#e8eaed',
        300: '#dadce0',
        400: '#bdc1c6',
        500: '#9aa0a6',
        600: '#80868b',
        700: '#5f6368',
        800: '#3c4043',
        900: '#202124',
      },

      // Акцентные цвета
      accent: {
        blue: '#1a73e8',
        green: '#34a853',
        yellow: '#fbbc04',
        red: '#ea4335',
        purple: '#9c27b0',
        orange: '#ff9800',
      },

      // Семантические цвета
      success: '#34a853',
      warning: '#fbbc04',
      error: '#ea4335',
      info: '#1a73e8',
```

```
    },

    fontFamily: {
      sans: ['Inter', 'sans-serif'],
      mono: ['JetBrains Mono', 'monospace'],
    },

    animation: {
      'fade-in': 'fadeIn 0.3s ease-in-out',
      'slide-up': 'slideUp 0.3s ease-out',
      'slide-down': 'slideDown 0.3s ease-out',
    }
  },

  shortcuts: {
    // Утилиты для быстрой стилизации
    'btn-primary': 'bg-primary-900 hover:bg-primary-800 text-white px-4 py-2 rounded-lg font-medium transition
    'btn-secondary': 'bg-primary-100 hover:bg-primary-200 text-primary-900 px-4 py-2 rounded-lg font-medium
    'card': 'bg-white dark:bg-primary-800 border border-primary-200 dark:border-primary-700 rounded-xl shado
    'input': 'border border-primary-300 dark:border-primary-600 rounded-lg px-3 py-2 focus:outline-none focus:r
  }
})
```

✅ Шаг 5: Настройка ESLint и Prettier

Создаем `.eslintrc.js`:

javascript

```
// .eslintrc.js
module.exports = {
  root: true,
  extends: ['@nuxt/eslint-config'],
  rules: {
    'vue/multi-word-component-names': 'off',
    'vue/no-v-html': 'off',
    '@typescript-eslint/no-unused-vars': ['error', {
      argsIgnorePattern: '^_',
      varsIgnorePattern: '^_'
    }],
  }
}
```

Создаем `.prettierrc`:

json

```
{
  "semi": false,
  "singleQuote": true,
  "tabWidth": 2,
  "trailingComma": "es5",
  "printWidth": 100,
  "bracketSpacing": true,
  "arrowParens": "avoid"
}
```

✅ Шаг 6: Настройка Husky и lint-staged

```bash
# Инициализация Husky
npx husky init

# Создаем pre-commit хук
echo "npx lint-staged" > .husky/pre-commit
```

Добавляем в `package.json`:

```json
{
  "scripts": {
    "build": "nuxt build",
    "dev": "nuxt dev",
    "generate": "nuxt generate",
    "preview": "nuxt preview",
    "postinstall": "nuxt prepare",
    "lint": "eslint .",
    "lint:fix": "eslint . --fix",
    "format": "prettier --write ."
  },
  "lint-staged": {
    "*.{js,ts,vue}": [
      "eslint --fix",
      "prettier --write"
    ],
    "*.{css,scss,json,md}": [
      "prettier --write"
    ]
  }
}
```

✅ Шаг 7: Создание структуры проекта

```
lynkr/
├── assets/
│   └── css/
│       └── main.css
├── components/
│   ├── ui/
│   │   ├── UiButton.vue
│   │   ├── UiInput.vue
│   │   ├── UiCard.vue
│   │   └── UiModal.vue
│   ├── auth/
│   │   ├── AuthLogin.vue
│   │   └── AuthRegister.vue
│   └── layout/
│       ├── AppHeader.vue
│       ├── AppSidebar.vue
│       └── ThemeToggle.vue
├── composables/
│   ├── useAuth.ts
│   ├── useTheme.ts
│   └── useSupabase.ts
├── layouts/
│   ├── default.vue
│   └── auth.vue
├── middleware/
│   └── auth.ts
├── pages/
│   ├── index.vue
│   ├── auth/
│   │   ├── login.vue
│   │   └── register.vue
│   └── dashboard.vue
├── plugins/
│   └── supabase.client.ts
├── stores/
│   ├── auth.ts
│   └── theme.ts
├── types/
│   ├── auth.ts
│   ├── database.ts
│   └── index.ts
├── utils/
│   ├── constants.ts
│   └── helpers.ts
```

```
└── server/
    └── api/
```

✅ Шаг 8: Настройка переменных окружения

Создаем `.env.example`:

```env
# Supabase
SUPABASE_URL=your_supabase_url
SUPABASE_ANON_KEY=your_supabase_anon_key

# App
NUXT_PUBLIC_APP_URL=http://localhost:3000
```

Создаем `.env` (скопировать из `.env.example` и заполнить)

✅ Шаг 9: Настройка Supabase

1. Создаем проект на supabase.com
2. Получаем URL и anon key
3. Создаем `plugins/supabase.client.ts`:

```typescript
// plugins/supabase.client.ts
import { createClient } from '@supabase/supabase-js'

export default defineNuxtPlugin(() => {
  const config = useRuntimeConfig()

  const supabase = createClient(
    config.public.supabaseUrl,
    config.public.supabaseAnonKey
  )

  return {
    provide: {
      supabase
    }
  }
})
```

✅ Шаг 10: Создание базовых типов

Создаем `types/database.ts`:

```typescript
```

```ts
// types/database.ts
export interface User {
  id: string
  email: string
  name: string
  avatar_url?: string
  settings?: UserSettings
  created_at: string
  updated_at: string
}

export interface UserSettings {
  theme: 'light' | 'dark' | 'system'
  defaultView: 'grid' | 'list'
  language: string
}

export interface Collection {
  id: string
  user_id: string
  name: string
  description?: string
  color?: string
  icon?: string
  parent_id?: string
  position: number
  created_at: string
  updated_at: string
}

export interface Link {
  id: string
  user_id: string
  collection_id: string
  url: string
  title: string
  description?: string
  color?: string
  is_favorite: boolean
  position: number
  metadata?: LinkMetadata
  created_at: string
  updated_at: string
}

export interface LinkMetadata {
```

```
  title?: string
  description?: string
  image?: string
  favicon?: string
  site_name?: string
}

export interface Tag {
  id: string
  user_id: string
  name: string
  color?: string
  created_at: string
}

export interface LinkTag {
  link_id: string
  tag_id: string
}
```

🗄️ Структура Базы Данных

SQL для создания таблиц в Supabase

sql

```sql
-- Включаем RLS (Row Level Security)
alter table auth.users enable row level security;

-- Таблица пользователей (расширение auth.users)
create table public.users (
  id uuid references auth.users on delete cascade primary key,
  email text unique not null,
  name text not null,
  avatar_url text,
  settings jsonb default '{}'::jsonb,
  created_at timestamptz default now(),
  updated_at timestamptz default now()
);

-- Таблица коллекций
create table public.collections (
  id uuid default gen_random_uuid() primary key,
  user_id uuid references public.users(id) on delete cascade not null,
  name text not null,
  description text,
  color text,
  icon text,
  parent_id uuid references public.collections(id) on delete cascade,
  position integer not null default 0,
  created_at timestamptz default now(),
  updated_at timestamptz default now()
);

-- Таблица ссылок
create table public.links (
  id uuid default gen_random_uuid() primary key,
  user_id uuid references public.users(id) on delete cascade not null,
  collection_id uuid references public.collections(id) on delete cascade not null,
  url text not null,
  title text not null,
  description text,
  color text,
  is_favorite boolean default false,
  position integer not null default 0,
  metadata jsonb default '{}'::jsonb,
  created_at timestamptz default now(),
  updated_at timestamptz default now()
);

-- Таблица тегов
create table public.tags (
```

```sql
  id uuid default gen_random_uuid() primary key,
  user_id uuid references public.users(id) on delete cascade not null,
  name text not null,
  color text,
  created_at timestamptz default now(),

  unique(user_id, name)
);

-- Связующая таблица ссылок и тегов
create table public.link_tags (
  link_id uuid references public.links(id) on delete cascade,
  tag_id uuid references public.tags(id) on delete cascade,
  primary key (link_id, tag_id)
);

-- Индексы для производительности
create index idx_collections_user_id on public.collections(user_id);
create index idx_collections_parent_id on public.collections(parent_id);
create index idx_links_user_id on public.links(user_id);
create index idx_links_collection_id on public.links(collection_id);
create index idx_tags_user_id on public.tags(user_id);

-- RLS политики
-- Пользователи могут видеть только свои данные
create policy "Users can view own data" on public.users
  for all using (auth.uid() = id);

create policy "Users can view own collections" on public.collections
  for all using (auth.uid() = user_id);

create policy "Users can view own links" on public.links
  for all using (auth.uid() = user_id);

create policy "Users can view own tags" on public.tags
  for all using (auth.uid() = user_id);

create policy "Users can view own link_tags" on public.link_tags
  for all using (auth.uid() = (select user_id from public.links where id = link_id));

-- Включаем RLS для всех таблиц
alter table public.users enable row level security;
alter table public.collections enable row level security;
alter table public.links enable row level security;
alter table public.tags enable row level security;
alter table public.link_tags enable row level security;
```

🎨 Базовая Дизайн-система

Создание базовых компонентов

components/ui/Button.vue

```vue
```

```
<template>
  <button
    :class="[
      'inline-flex items-center justify-center font-medium transition-colors',
      'focus:outline-none focus:ring-2 focus:ring-offset-2',
      'disabled:opacity-50 disabled:cursor-not-allowed',
      sizeClasses,
      variantClasses,
      className
    ]"
    :disabled="disabled"
    v-bind="$attrs"
  >
    <slot />
  </button>
</template>

<script setup lang="ts">
interface Props {
  variant?: 'primary' | 'secondary' | 'ghost' | 'outline'
  size?: 'sm' | 'md' | 'lg'
  disabled?: boolean
  className?: string
}

const props = withDefaults(defineProps<Props>(), {
  variant: 'primary',
  size: 'md',
  disabled: false,
  className: ''
})

const sizeClasses = computed(() => {
  const sizes = {
    sm: 'px-3 py-1.5 text-sm rounded-md',
    md: 'px-4 py-2 text-sm rounded-lg',
    lg: 'px-6 py-3 text-base rounded-lg'
  }
  return sizes[props.size]
})

const variantClasses = computed(() => {
  const variants = {
    primary: 'bg-primary-900 hover:bg-primary-800 text-white focus:ring-primary-500',
    secondary: 'bg-primary-100 hover:bg-primary-200 text-primary-900 focus:ring-primary-500',
    ghost: 'hover:bg-primary-100 text-primary-900 focus:ring-primary-500',
```

```
    outline: 'border border-primary-300 hover:bg-primary-50 text-primary-900 focus:ring-primary-500'
  }
  return variants[props.variant]
})
</script>
```

🔄 Следующие шаги

После завершения настройки проекта переходим к:

1. **Создание системы авторизации** (Login/Register формы)
2. **Настройка middleware для защищенных роутов**
3. **Создание базового layout с навигацией**
4. **Тестирование базовой функциональности**

📝 Коммиты для этой фазы

```bash
# Примерные коммиты
git add .
git commit -m "feat: initial Nuxt 3 setup with TypeScript"

git add .
git commit -m "feat: configure UnoCSS and design system"

git add .
git commit -m "feat: setup Supabase and database schema"

git add .
git commit -m "feat: add basic UI components and structure"

git add .
git commit -m "feat: configure ESLint, Prettier, and Husky"
```

✅ Критерии завершения Фазы 1

- [ ] Проект запускается без ошибок (`npm run dev`)
- [ ] TypeScript работает корректно
- [ ] Supabase подключен и настроен
- [ ] Базовые UI компоненты созданы
- [ ] Структура проекта организована

- [ ] Линтеры и форматтеры настроены
- [ ] База данных создана в Supabase
- [ ] Все изменения зафиксированы в Git