Frontend Setup Guide for BuildersHub Public API

Available API Endpoints

Endpoint	Method	Description
`/api/public/tenant`	GET	Get builder/tenant info
`/api/public/communities`	GET	List all communities
`/api/public/communities/[slug]`	GET	Get single community
`/api/public/homes`	GET	List all homes
`/api/public/homes/[slug]`	GET	Get single home
`/api/public/floorplans`	GET	List all floorplans
`/api/public/floorplans/[slug]`	GET	Get single floorplan
`/api/public/pages`	GET	List CMS pages
`/api/public/pages/[slug]`	GET	Get single page
`/api/public/inquiries`	POST	Submit contact form

Note: Market areas, blog, categories, and tags endpoints don't exist yet. Communities include market area data in their response.

Step 1: Create Your Frontend Project

```
# Using Next.js (recommended)
npx create-next-app@latest my-builder-site --typescript --tailwind --app

# Or using Vite + React
npm create vite@latest my-builder-site -- --template react-ts
```

Step 2: Install Dependencies

```
cd my-builder-site
npm install @tanstack/react-query axios
```

Step 3: Configure Environment Variables

Create `.env.local`:

```
# Your BuildersHub API base URL
NEXT_PUBLIC_API_URL=https://ridgelinehomes.forgehome.io
```

```
# Your API key (get from super-admin panel)
NEXT_PUBLIC_API_KEY=fh_pk_xxxxxxxxxxxxxxxxxxxx
```

Step 4: Create API Client

Create `lib/api.ts` :

```
import axios from "axios";

const API_URL = process.env.NEXT_PUBLIC_API_URL || "";
const API_KEY = process.env.NEXT_PUBLIC_API_KEY || "";

export const api = axios.create({
  baseURL: API_URL,
  headers: {
    "Content-Type": "application/json",
    "X-API-Key": API_KEY,
  },
});

// Types
export interface Community {
  id: string;
  name: string;
  slug: string;
  marketingHeadline: string | null;
  showMarketingHeadline: boolean;
  headline: string | null;
  description: string | null;
  address: string | null;
  city: string | null;
  county: string | null;
  state: string | null;
  zipCode: string | null;
  latitude: number | null;
  longitude: number | null;
  phoneNumber: string | null;
  salesOfficeHours: any;
  gallery: string[];
  videoUrl: string | null;
  interactiveSiteMap: string | null;
  communityMap: string | null;
  schoolDistrict: string | null;
  elementarySchool: string | null;
  middleSchool: string | null;
  highSchool: string | null;
  amenities: string[];
  status: string;
  priceMin: number | null;
```

```typescript
  priceMax: number | null;
  sqftMin: number | null;
  sqftMax: number | null;
  directions: string | null;
  marketArea: {
    id: string;
    name: string;
    slug: string;
    state: string | null;
    city: string | null;
  } | null;
  seo: {
    title: string | null;
    description: string | null;
    keywords: string[];
    ogTitle: string | null;
    ogDescription: string | null;
    ogImage: string | null;
  } | null;
  salesTeams: Array<{
    isPrimary: boolean;
    displayOrder: number;
    salesTeam: {
      id: string;
      name: string;
      title: string | null;
      email: string | null;
      phone: string | null;
      photo: string | null;
    };
  }>;
  _count: {
    homes: number;
  };
}

export interface Home {
  id: string;
  name: string;
  slug: string;
  lotNumber: string | null;
  address: string | null;
  street: string | null;
  city: string | null;
  state: string | null;
  zipCode: string | null;
  price: number | null;
  bedrooms: number | null;
  bathrooms: number | null;
  squareFeet: number | null;
  stories: number | null;
  garages: number | null;
  status: string;
  marketingHeadline: string | null;
  showMarketingHeadline: boolean;
  gallery: string[];
  openHouseDate: string | null;
  availableDate: string | null;
  estimatedCompletion: string | null;
  latitude: number | null;
  longitude: number | null;
  community: {
    id: string;
    name: string;
    slug: string;
    city: string | null;
    state: string | null;
  } | null;
  floorplan: {
    id: string;
    name: string;
    slug: string;
  } | null;
}

export interface Floorplan {
  id: string;
  name: string;
  slug: string;
  modelNumber: string | null;
  planTypes: string[];
  baseBedrooms: number | null;
  baseBathrooms: number | null;
  baseSquareFeet: number | null;
  baseStories: number | null;
  baseGarages: number | null;
  basePrice: number | null;
  marketingHeadline: string | null;
  showMarketingHeadline: boolean;
  headline: string | null;
  description: string | null;
  features: string[];
  gallery: string[];
  elevationGallery: string[];
  plansImages: string[];
  virtualTourUrl: string | null;
  videoUrl: string | null;
  brochureUrl: string | null;
  interactivePlanUrl: string | null;
  communityFloorplans: Array<{
    price: number | null;
    bedrooms: number | null;
    bathrooms: number | null;
    squareFeet: number | null;
    community: {
      id: string;
      name: string;
      slug: string;
    };
  }>;
  _count: {
    homes: number;
  };
}

export interface Tenant {
  id: string;
  name: string;
  slug: string;
  description: string | null;
  logo: string | null;
  favicon: string | null;
  builderName: string | null;
  builderEmail: string | null;
  builderPhone: string | null;
  builderAddress: string | null;
  builderCity: string | null;
  builderState: string | null;
  builderZip: string | null;
  facebookUrl: string | null;
  twitterUrl: string | null;
  instagramUrl: string | null;
  linkedinUrl: string | null;
  youtubeUrl: string | null;
  promoBannerEnabled: boolean;
  promoBannerDescription: string | null;
  promoBannerLink: string | null;
}

// API Functions
export const fetchTenant = () => api.get<Tenant>("/api/public/tenant");

export const fetchCommunities = (params?: {
  status?: string;
  marketAreaId?: string;
  limit?: number;
}) => api.get<Community[]>("/api/public/communities", { params });

export const fetchCommunity = (slug: string) =>
  api.get<Community>(`/api/public/communities/${slug}`);

export const fetchHomes = (params?: {
  status?: string;
  communityId?: string;
  communitySlug?: string;
  minPrice?: number;
  maxPrice?: number;
  bedrooms?: number;
  limit?: number;
  featured?: boolean;
}) => api.get<Home[]>("/api/public/homes", { params });

export const fetchHome = (slug: string) =>
  api.get<Home>(`/api/public/homes/${slug}`);

export const fetchFloorplans = (params?: {
  communityId?: string;
  communitySlug?: string;
  minBedrooms?: number;
  maxPrice?: number;
  planType?: string;
  limit?: number;
}) => api.get<Floorplan[]>("/api/public/floorplans", { params });

export const fetchFloorplan = (slug: string) =>
  api.get<Floorplan>(`/api/public/floorplans/${slug}`);

export const submitInquiry = (data: {
  type: string;
  firstName: string;
  lastName: string;
  email: string;
  phone?: string;
  message?: string;
  communityId?: string;
  homeId?: string;
  floorplanId?: string;
}) => api.post("/api/public/inquiries", data);
```

Step 5: Set Up React Query Provider

Create `providers/query-provider.tsx`:

```
"use client";

import { QueryClient, QueryClientProvider } from "@tanstack/react-query";
import { useState } from "react";

export function QueryProvider({ children }: { children: React.ReactNode }) {
  const [queryClient] = useState(
    () =>
      new QueryClient({
        defaultOptions: {
          queries: {
            staleTime: 60 * 1000, // 1 minute
            refetchOnWindowFocus: false,
          },
        },
      })
  );

  return (
    <QueryClientProvider client={queryClient}>{children}</QueryClientProvider>
  );
}
```

Update `app/layout.tsx` :

```
import { QueryProvider } from "@/providers/query-provider";

export default function RootLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  return (
    <html lang="en">
      <body>
        <QueryProvider>{children}</QueryProvider>
      </body>
    </html>
  );
}
```

Step 6: Create Custom Hooks

Create `hooks/use-api.ts` :

```
import { useQuery, useMutation } from "@tanstack/react-query";
import {
  fetchTenant,
  fetchCommunities,
  fetchCommunity,
  fetchHomes,
  fetchHome,
  fetchFloorplans,
  fetchFloorplan,
  submitInquiry,
} from "@/lib/api";

// Tenant/Builder Info
export function useTenant() {
  return useQuery({
    queryKey: ["tenant"],
    queryFn: async () => {
      const { data } = await fetchTenant();
      return data;
    },
  });
}

// Communities
export function useCommunities(params?: Parameters<typeof fetchCommunities>[0]) {
  return useQuery({
    queryKey: ["communities", params],
    queryFn: async () => {
      const { data } = await fetchCommunities(params);
      return data;
    },
  });
}

export function useCommunity(slug: string) {
  return useQuery({
    queryKey: ["community", slug],
    queryFn: async () => {
      const { data } = await fetchCommunity(slug);
      return data;
    },
    enabled: !!slug,
  });
}

// Homes
export function useHomes(params?: Parameters<typeof fetchHomes>[0]) {
  return useQuery({
    queryKey: ["homes", params],
    queryFn: async () => {
      const { data } = await fetchHomes(params);
      return data;
    },
  });
}

export function useHome(slug: string) {
  return useQuery({
    queryKey: ["home", slug],
    queryFn: async () => {
      const { data } = await fetchHome(slug);
      return data;
    },
    enabled: !!slug,
  });
}

// Floorplans
export function useFloorplans(params?: Parameters<typeof fetchFloorplans>[0]) {
  return useQuery({
    queryKey: ["floorplans", params],
    queryFn: async () => {
      const { data } = await fetchFloorplans(params);
      return data;
    },
  });
}

export function useFloorplan(slug: string) {
  return useQuery({
    queryKey: ["floorplan", slug],
    queryFn: async () => {
      const { data } = await fetchFloorplan(slug);
      return data;
    },
    enabled: !!slug,
  });
}

// Contact Form
export function useSubmitInquiry() {
  return useMutation({
    mutationFn: submitInquiry,
  });
}
```

Step 7: Example Components

Communities List Page

```
// app/communities/page.tsx
"use client";

import { useCommunities } from "@/hooks/use-api";
import Link from "next/link";

export default function CommunitiesPage() {
  const { data: communities, isLoading, error } = useCommunities();

  if (isLoading) return <div>Loading communities...</div>;
  if (error) return <div>Error loading communities</div>;

  return (
    <div className="container mx-auto py-8">
      <h1 className="text-3xl font-bold mb-8">Our Communities</h1>
      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
        {communities?.map((community) => (
          <Link
            key={community.id}
            href={`/communities/${community.slug}`}
            className="block border rounded-lg overflow-hidden hover:shadow-lg
transition"
          >
            {community.gallery?.[0] && (
              <img
                src={community.gallery[0]}
                alt={community.name}
                className="w-full h-48 object-cover"
              />
            )}
            <div className="p-4">
              <h2 className="text-xl font-semibold">{community.name}</h2>
              <p className="text-gray-600">
                {community.city}, {community.state}
              </p>
              {community.priceMin && (
                <p className="text-lg font-medium mt-2">
                  From ${community.priceMin.toLocaleString()}
                </p>
              )}
              <p className="text-sm text-gray-500 mt-1">
                {community._count.homes} homes available
              </p>
            </div>
          </Link>
        ))}
      </div>
    </div>
  );
}
```

Single Community Page

```
// app/communities/[slug]/page.tsx
"use client";

import { useCommunity, useHomes, useFloorplans } from "@/hooks/use-api";
import { useParams } from "next/navigation";

export default function CommunityPage() {
  const params = useParams();
  const slug = params.slug as string;

  const { data: community, isLoading } = useCommunity(slug);
  const { data: homes } = useHomes({ communitySlug: slug });
  const { data: floorplans } = useFloorplans({ communitySlug: slug });

  if (isLoading) return <div>Loading...</div>;
  if (!community) return <div>Community not found</div>;

  return (
    <div className="container mx-auto py-8">
      {/* Hero */}
      <div className="relative h-96 mb-8">
        {community.gallery?.[0] && (
          <img
            src={community.gallery[0]}
            alt={community.name}
            className="w-full h-full object-cover rounded-lg"
          />
        )}
        <div className="absolute bottom-0 left-0 right-0 bg-gradient-to-t from-
black/70 p-6">
          <h1 className="text-4xl font-bold text-white">{community.name}</h1>
          <p className="text-white/80">
            {community.city}, {community.state}
          </p>
        </div>
      </div>

      {/* Description */}
      {community.description && (
        <p className="text-lg mb-8">{community.description}</p>
      )}

      {/* Schools */}
      {(community.elementarySchool || community.middleSchool ||
community.highSchool) && (
        <div className="mb-8">
          <h2 className="text-2xl font-semibold mb-4">Schools</h2>
          <ul className="space-y-2">
            {community.elementarySchool && (
              <li>Elementary: {community.elementarySchool}</li>
            )}
            {community.middleSchool && (
              <li>Middle: {community.middleSchool}</li>
            )}
            {community.highSchool && (
              <li>High School: {community.highSchool}</li>
            )}
          </ul>
        </div>
      )}

      {/* Amenities */}
      {community.amenities?.length > 0 && (
        <div className="mb-8">
          <h2 className="text-2xl font-semibold mb-4">Amenities</h2>
          <div className="flex flex-wrap gap-2">
            {community.amenities.map((amenity) => (
              <span
                key={amenity}
                className="px-3 py-1 bg-gray-100 rounded-full text-sm"
              >
                {amenity}
              </span>
            ))}
          </div>
        </div>
      )}

      {/* Available Homes */}
      <div className="mb-8">
        <h2 className="text-2xl font-semibold mb-4">
          Available Homes ({homes?.length || 0})
        </h2>
        <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
          {homes?.map((home) => (
            <div key={home.id} className="border rounded-lg p-4">
              <h3 className="font-semibold">{home.name}</h3>
              <p className="text-lg font-medium">
                ${home.price?.toLocaleString()}
              </p>
              <p className="text-sm text-gray-600">
                {home.bedrooms} bed | {home.bathrooms} bath |{" "}
                {home.squareFeet?.toLocaleString()} sqft
              </p>
            </div>
          ))}
        </div>
      </div>

      {/* Floorplans */}
      <div>
        <h2 className="text-2xl font-semibold mb-4">
          Floorplans ({floorplans?.length || 0})
        </h2>
        <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
          {floorplans?.map((plan) => (
            <div key={plan.id} className="border rounded-lg p-4">
              <h3 className="font-semibold">{plan.name}</h3>
              <p className="text-lg font-medium">
                From ${plan.basePrice?.toLocaleString()}
              </p>
              <p className="text-sm text-gray-600">
                {plan.baseBedrooms} bed | {plan.baseBathrooms} bath |{" "}
                {plan.baseSquareFeet?.toLocaleString()} sqft
              </p>
            </div>
          ))}
        </div>
      </div>
    </div>
```

```
  );
}
```

Contact Form

```
// components/contact-form.tsx
"use client";

import { useState } from "react";
import { useSubmitInquiry } from "@/hooks/use-api";

interface ContactFormProps {
  communityId?: string;
  homeId?: string;
  floorplanId?: string;
}

export function ContactForm({ communityId, homeId, floorplanId }: ContactFormProps)
{
  const [formData, setFormData] = useState({
    firstName: "",
    lastName: "",
    email: "",
    phone: "",
    message: "",
  });

  const { mutate, isPending, isSuccess, isError } = useSubmitInquiry();

  const handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    mutate({
      type: "GENERAL",
      ...formData,
      communityId,
      homeId,
      floorplanId,
    });
  };

  if (isSuccess) {
    return (
      <div className="p-4 bg-green-100 text-green-800 rounded-lg">
        Thank you! We'll be in touch soon.
      </div>
    );
  }

  return (
    <form onSubmit={handleSubmit} className="space-y-4">
      <div className="grid grid-cols-2 gap-4">
```

```
          <input
            type="text"
            placeholder="First Name"
            required
            className="border rounded px-3 py-2"
            value={formData.firstName}
            onChange={(e) => setFormData({ ...formData, firstName: e.target.value })}
          />
          <input
            type="text"
            placeholder="Last Name"
            required
            className="border rounded px-3 py-2"
            value={formData.lastName}
            onChange={(e) => setFormData({ ...formData, lastName: e.target.value })}
          />
        </div>
        <input
          type="email"
          placeholder="Email"
          required
          className="w-full border rounded px-3 py-2"
          value={formData.email}
          onChange={(e) => setFormData({ ...formData, email: e.target.value })}
        />
        <input
          type="tel"
          placeholder="Phone"
          className="w-full border rounded px-3 py-2"
          value={formData.phone}
          onChange={(e) => setFormData({ ...formData, phone: e.target.value })}
        />
        <textarea
          placeholder="Message"
          rows={4}
          className="w-full border rounded px-3 py-2"
          value={formData.message}
          onChange={(e) => setFormData({ ...formData, message: e.target.value })}
        />
        {isError && (
          <p className="text-red-600">Something went wrong. Please try again.</p>
        )}
        <button
          type="submit"
          disabled={isPending}
          className="w-full bg-blue-600 text-white py-2 rounded hover:bg-blue-700
disabled:opacity-50"
        >
          {isPending ? "Sending..." : "Submit"}
        </button>
      </form>
```

```
  );
}
```

Step 8: Query Parameters Reference

Communities

Parameter	Type	Description
`status`	string	Filter by status
`marketAreaId`	string	Filter by market area
`limit`	number	Max results to return

Homes

Parameter	Type	Description
`status`	string	AVAILABLE, COMING_SOON, RESERVED, SOLD
`communityId`	string	Filter by community ID
`communitySlug`	string	Filter by community slug
`minPrice`	number	Minimum price
`maxPrice`	number	Maximum price
`bedrooms`	number	Minimum bedrooms
`limit`	number	Max results
`featured`	boolean	Featured homes only

Floorplans

Parameter	Type	Description
`communityId`	string	Filter by community ID
`communitySlug`	string	Filter by community slug
`minBedrooms`	number	Minimum bedrooms
`maxPrice`	number	Maximum base price
`planType`	string	Filter by plan type
`limit`	number	Max results

Step 9: Run Your Frontend

```
npm run dev
```

Your frontend should now be able to consume all available BuildersHub API endpoints. The market areas data is included in community responses under the `marketArea` field.

Troubleshooting

CORS Errors

If you encounter CORS errors:

1. Ensure your frontend is running on an allowed origin (localhost:3000, 3001, 3002, 4000, or 5173)
2. Add custom origins via `ALLOWED_CORS_ORIGINS` environment variable on the BuildersHub backend
3. Vercel and Netlify preview deployments are automatically allowed

API Key Issues

- API keys must start with `fh_pk_`
- Get your API key from the super-admin panel under the tenant's API Keys section
- API keys cannot be retrieved after creation - save them securely

404 Errors

- Ensure you're using the correct endpoint paths (e.g., `/api/public/homes` not `/api/properties`)
- Check that the tenant subdomain in your API URL is correct