

LET'S BUILD A 0-COST INVITE-ONLY WEBSITE WITH NEXT.JS AND AIRTABLE!

Luciano Mammino (@loige)

loige.link/micro42

CONF42



META_SLIDE!
↻

loige.link/micro42



OUR MISSION TODAY:

LET'S BUILD AN INVITE-ONLY WEBSITE!

15 SECONDS DEMO

secret-pizza-party: invites - AI x

airtable.com/app/Nh2Jt9DhAaKike/tblmYeU8C2e1yWugM/viw4peHu8szS...

secret-pizza-party

invites

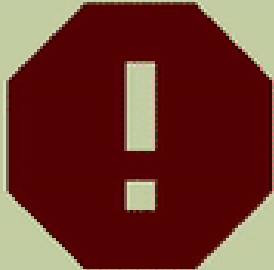
Views | Grid view

	A invite	A name	A favouriteColor
1	14b25700-fe5b-45e8-a9be-4863b6239fcf	Leonardo	blue
2	ce7d5886-2166-4aee-8038-648f68b9739d	Michelangelo	orange
3	ef7ab7b7-33d5-43b9-ad73-e73bb8fd8e77	Raffaello	red
4	b0cbb4a4-8a31-4bc1-bee9-d6fe39c1a6b3	Donatello	purple

4 records

Airtable API - secret-pizz... | Submit report to Airtable | Secret Pizza Party





secret-pizza-party-fgypfb66-lmammino.vercel.app



No code provided

loige5

(SELF-IMPOSED) REQUIREMENTS

-  Iterate quickly
-  Simple to host, maintain and update
- Lightweight backend
-  Non-techy people can easily access the data
-  Cheap (or even FREE) hosting!

LET ME INTRODUCE MYSELF FIRST...

👋 I'm Luciano (🇮🇹🍕🍝🗑️)



💻 Senior Architect @ fourTheorem (Dublin 🇮🇹)

📖 Co-Author of Node.js Design Patterns 🙌

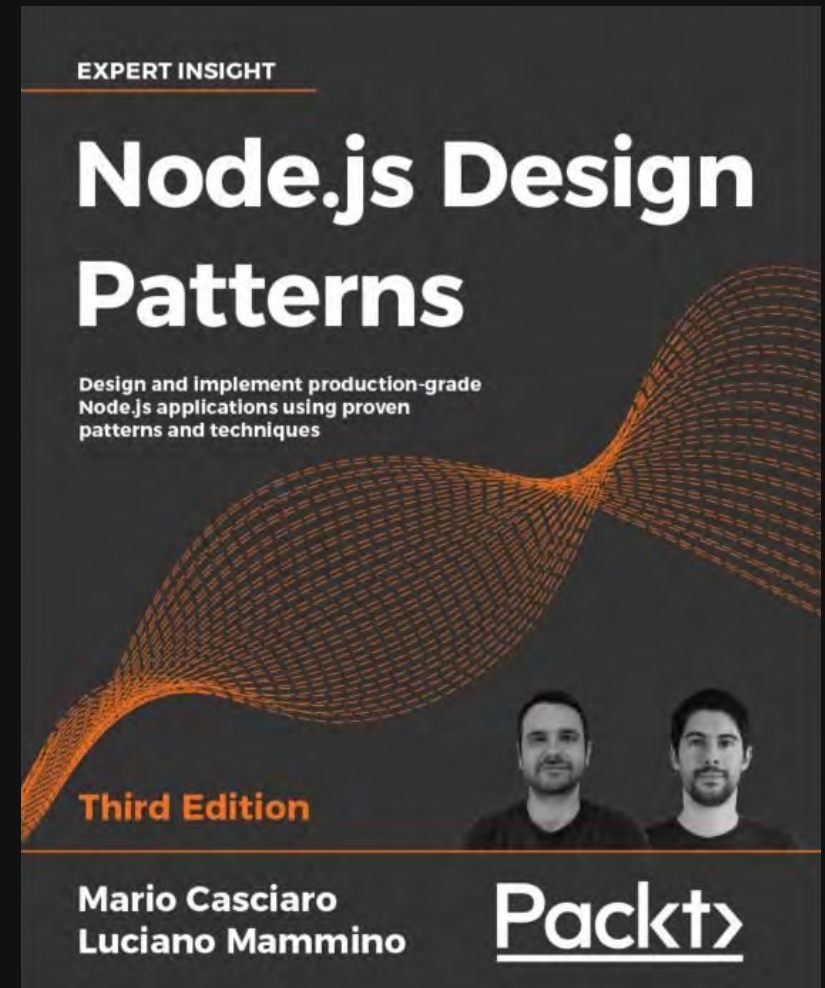
Let's connect!

loige.co (blog)

[@loige](https://twitter.com/loige) (twitter)

[loige](https://www.twitch.tv/loige) (twitch)

[lmammino](https://github.com/lmammino) (github)



nodejsdp.link



ALWAYS RE-IMAGINING

WE ARE A PIONEERING TECHNOLOGY CONSULTANCY FOCUSED ON AWS AND SERVERLESS

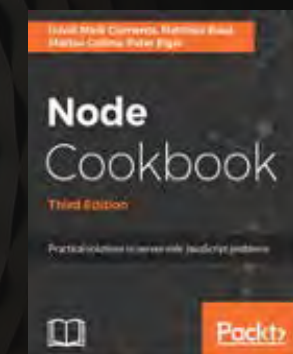
Accelerated Serverless | AI as a Service | Platform Modernisation

✉ Reach out to us at hello@fourTheorem.com

👤 We are always looking for talent: fth.link/careers



Select
Consulting
Partner





AGENDA

- Choosing the tech stack
- The data flow
- Using *Airtable* as a database
- Creating APIs with Next.js and Vercel
- Creating custom React Hooks
- Using user interaction to update the data
- Security considerations

TECH STACK 🥞

NEXT.JS



▲ Vercel

 Airtable

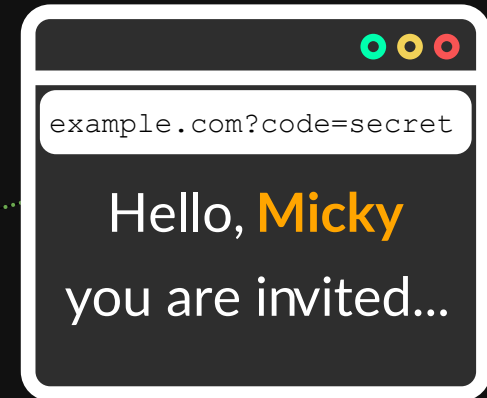
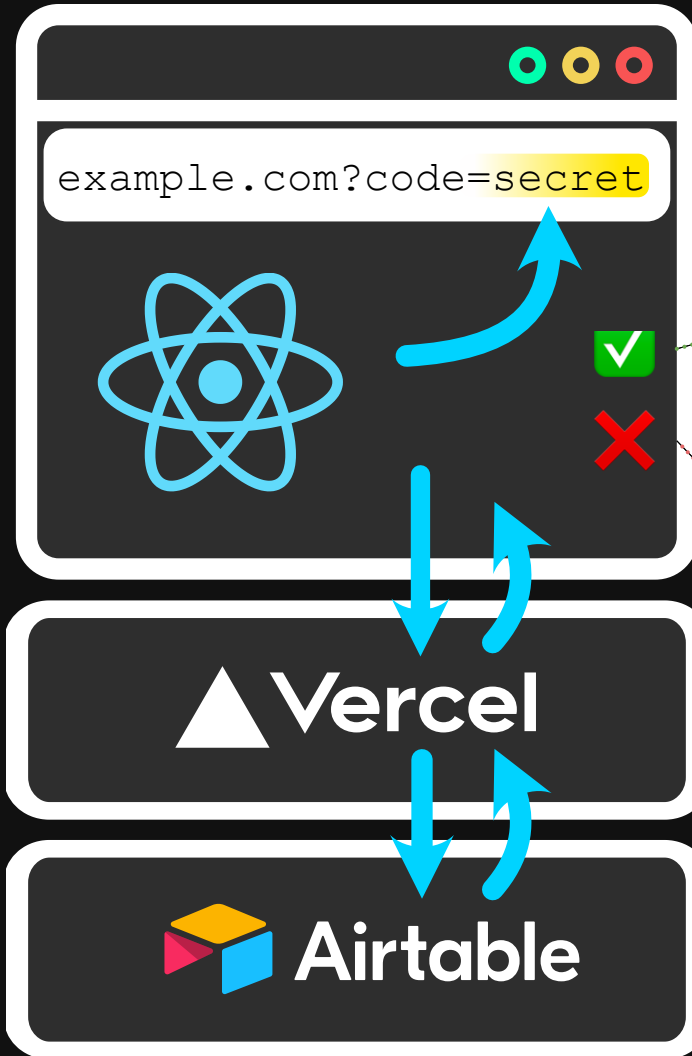
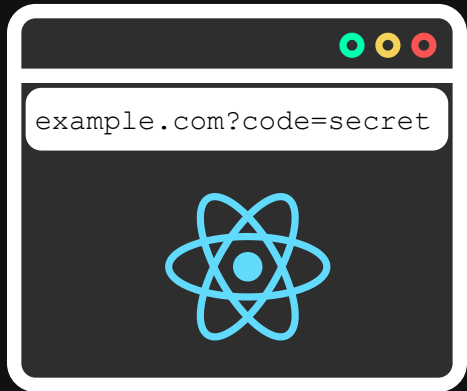
MAKING A NEXT.JS PRIVATE

- Every guest should see something different
- People without an invite code should not be able to access any content

1 Load React SPA

2 Code validation

3 View invite (or error)



STEP 1.

LET'S ORGANIZE THE DATA IN AIRTABLE

MANAGING DATA

- Invite codes are UUIDs
- Every record contains the information for every guest (name, etc)

The screenshot shows a web application interface for 'secret-pizza-party'. The top navigation bar is green and contains the app name and tabs for 'Data', 'Automations', and 'Interfaces'. Below this, there's a sub-navigation bar with 'invites' selected and an 'Add or import' button. A toolbar below the navigation bar includes options for 'Views', 'Grid view', 'Hide fields', 'Filter', 'Group', 'Sort', 'Color', and 'Share view'. On the left, there's a search box for views and a 'Grid view' option selected. The main content area displays a table with the following data:

	A invite	A name	A favouriteColor	A weapon
1	14b25700-fe5b-45e8-a9be-4863b6239fcf	Leonardo	blue	Twin Katana
2	ce7d5886-2166-4aee-8038-548f68b9739d	Michelangelo	orange	Nunchaku
3	ef7ab7b7-33d5-43b9-ad73-e73bb8fd8e77	Raffaello	red	Twin Sai
4	b0cbb4a4-8a31-4bc1-bee9-d6fe39c1a6b3	Donatello	purple	Bo
+				

The bottom left corner features the 'loige' logo, and the bottom right corner shows the page number '14'.

AIRTABLE LINGO

The image shows a screenshot of an Airtable workspace named "secret-pizza-party". The interface includes a top navigation bar with "Data", "Automations", and "Interfaces" tabs. Below this is a table view with a header row and four data rows. Annotations with arrows point to various parts of the interface:

- Base (project)**: Points to the "secret-pizza-party" header.
- Table**: Points to the "invites" dropdown menu.
- Fields**: Points to the column headers "name", "favouriteColor", and "weapon".
- Records**: Points to the individual rows of data in the table.

	A invite	A name	A favouriteColor	A weapon
1	14b25700-fe5b-45e8-a9be-4863b6239fcf	Leonardo	blue	Twin Katana
2	ce7d5886-2166-4aee-8038-548f68b9739d	Michelangelo	orange	Nunchaku
3	ef7ab7b7-33d5-43b9-ad73-e73bb8fd8e77	Raffaello	red	Twin Sai
4	b0cbb4a4-8a31-4bc1-bee9-d6fe39c1a6b3	Donatello	purple	Bo

STEP 2.

NEXT.JS SCAFFOLDING AND RETRIEVING INVITES

NEW NEXT.JS PROJECTS

```
npx create-next-app@12.2 --typescript --use-npm
```

(used Next.js 12.2)

INVITE TYPE

```
export interface Invite {  
  code: string,  
  name: string,  
  favouriteColor: string,  
  weapon: string,  
  coming?: boolean,  
}
```

AIRTABLE SDK

```
npm i --save airtable
```

```
export AIRTABLE_API_KEY="put your api key here"  
export AIRTABLE_BASE_ID="put your base id here"
```

- INTRODUCTION
- METADATA
- RATE LIMITS
- AUTHENTICATION
- INVITES TABLE
- Fields
- List records
- Retrieve a record
- Create records
- Update records
- Delete records
- ERRORS

INVITES TABLE

The id for `invites` is `tblmYeU8C2e1yWugM`. Table ids and table names can be used interchangeably in API requests. Using table ids means table name changes do not require modifications to your API request.

Fields

Each record in the `invites` table contains the following fields:

Field names and field ids can be used interchangeably. Using field ids means field name changes do not require modifications to your API request. We recommend using field ids over field names where possible, to reduce modifications to your API request if the user changes the field name later.

FIELD NAME	FIELD ID	TYPE	DESCRIPTION
<code>invite</code>	<code>fld3TJzEKvkvh8ICS</code>	Text	<code>string</code> A single line of text.
<code>name</code>	<code>fld6XKJjoYtQJCEP7</code>	Text	<code>string</code> A single line of text.
<code>favouriteColor</code>	<code>fld7kaVe0xYhAozSr</code>	Text	<code>string</code> A single line of text.
<code>weapon</code>	<code>fldYrpeu9tivIwvIM</code>	Text	<code>string</code> A single line of text.

List invites records

To list records in `invites`, issue a `GET` request to the `invites` endpoint. Note that table names and table ids can be used interchangeably. Using table ids means table name changes do not require modifications to your API request.

Returned records do not include any fields with "empty" values, e.g. `"`, `[]`, or `false`.

You can filter, sort, and format the results with the following query parameters. Note that these parameters need to be URL encoded. You can use our [API URL encoder tool](#) to help with this. If you are using a helper library like [Airtable.js](#), these parameters will be automatically encoded.

fields optional
array of strings
Only data for fields whose names are in this list will be included in the result. If you don't need every field, you can use this parameter to reduce the amount of data transferred.

For example, to only return data from `invite` and `name`, send these two query parameters:

```
fields%5B%5D=invite&fields%5B%5D=name
```

curl JavaScript

EXAMPLE VALUES

```
"14b25700-fe5b-45e8-a9be-4863b6239fc!"
"ce7d588e-2166-4aee-8038-548f68b9739d"
"ef7ab7b7-33d5-43b9-ad73-e73bb0fd8e77"
"b0cbb4a4-8a31-4bc1-bee9-d6fe39c1a6b3"
```

EXAMPLE VALUES

```
"Leonardo"
"Michelangelo"
"Raffaello"
"Donatello"
```

EXAMPLE VALUES

```
"blue"
"orange"
"red"
"purple"
```

EXAMPLE VALUES

```
"Twin Katana"
"Nunchaku"
"Twin Sai"
"Bo"
```

EXAMPLE REQUEST

```
curl "https://api.airtable.com/v0/appNk23t90hAaKike/invites?maxRecords=35&view=Grid%20view" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

EXAMPLE RESPONSE

```
{
  "records": [
    {
      "id": "recAMNFj8TMeZx7b3",
      "createdAt": "2022-07-15T20:01:51.000Z",
      "fields": {
        "invite": "14b25700-fe5b-45e8-a9be-4863b6239fc!",
        "name": "Leonardo",
        "favouriteColor": "blue",
        "weapon": "Twin Katana"
      }
    }
  ]
}
```



INTRODUCTION

METADATA

RATE LIMITS

AUTHENTICATION

INVITES TABLE

Fields

List records

Retrieve a record

Create records

Update records

Delete records

ERRORS

List invites records

To list records in `invites`, use the `select` method.

`select` returns a query object. To fetch the records matching that query, use the `eachPage` or `firstPage` method of the query object.

Returned records do not include any fields with "empty" values, e.g. "", [], or `false`.

You can use the following parameters to filter, sort, and format the results:

fields Only data for fields whose names are in this list will be included in the result. If you don't need every field, you can use this parameter to reduce the amount of data transferred.

`array of strings`
optional

For example, to only return data from `invite` and `name`, pass in:

```
fields: ["invite", "name"]
```

You can also perform the same action with field ids (they can be found in the fields section):

```
fields: ["fld3TJzEKvkvh8IC5", "fld6XKJjoYtQJECp7"]
```

filterByFormula A `formula` used to filter records. The formula will be evaluated for each record, and if the result is not `0`, `false`, "", `NaN`, [], or `#Error!` the record will be included in the response. We recommend testing your formula in the Formula field UI before using it in your API request.

`string`
optional

If combined with the `view` parameter, only records in that view which satisfy the formula will be returned.

The formula must be encoded first before passing it as a value. You can use [this tool](#) to not only encode the formula but also create the entire url you need. For example, to only include records where `invite` isn't empty, pass in `NOT({invite} = '')` as a parameter like this:

curl

JavaScript

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey: 'YOUR_API_KEY'}).base('appNh2Jt9DhAaKike');

base('invites').select({
  // Selecting the first 3 records in Grid view:
  maxRecords: 3,
  view: "Grid view"
}).eachPage(function page(records, fetchNextPage) {
  // This function ('page') will get called for each page of records.

  records.forEach(function(record) {
    console.log('Retrieved', record.get('invite'));
  });

  // To fetch the next page of records, call 'fetchNextPage'.
  // If there are more records, 'page' will get called again.
  // If there are no more records, 'done' will get called.
  fetchNextPage();
}, function done(err) {
  if (err) { console.error(err); return; }
});
```

OUTPUT

```
Retrieved 14b25700-fe5b-45e8-a9be-4863b6239fcf
Retrieved ce7d5886-2166-4aee-8038-548f68b9739d
Retrieved ef7ab7b7-33d5-43b9-ad73-e73bb8fd8e77
```

FETCH FIRST PAGE

```
// If you only want the first page of records, you can
// use 'firstPage' instead of 'eachPage'.
base('invites').select({
  view: 'Grid view'
}).firstPage(function(err, records) {
  if (err) { console.error(err); return; }
  records.forEach(function(record) {
```

matching that query, use the `eachPage` or `firstPage`

" values, e.g. "", [], or `false`.

and format the results:

those names are in this list will be included in the
 ed every field, you can use this parameter to reduce
 transferred.

return data from `invite` and `name`, pass in:

```
["name"]
```

the same action with field ids (they can be found in

```
["zEKvkvh8IC5", "fld6XKJjoYtQJECp7"]
```

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey: 'YOUR_API_KEY'}).base('appNh2Jt9DhAaKike');

base('invites').select({
  // Selecting the first 3 records in Grid view:
  maxRecords: 3,
  view: "Grid view"
}).eachPage(function page(records, fetchNextPage) {
  // This function ('page') will get called for each page of records.

  records.forEach(function(record) {
    console.log('Retrieved', record.get('invite'));
  });

  // To fetch the next page of records, call 'fetchNextPage'.
  // If there are more records, 'page' will get called again.
  // If there are no more records, 'done' will get called.
  fetchNextPage();

}, function done(err) {
  if (err) { console.error(err); return; }
});
```

```
1 // utils/airtable.ts
2
3 import Airtable from 'airtable'
4 import { Invite } from '../types/invite'
5
6 if (!process.env.AIRTABLE_API_KEY) {
7   throw new Error('AIRTABLE_API_KEY is not set')
8 }
9 if (!process.env.AIRTABLE_BASE_ID) {
10  throw new Error('AIRTABLE_BASE_ID is not set')
11 }
12
13 const airtable = new Airtable({ apiKey: process.env.AIRTABLE_API_KEY })
14 const base = airtable.base(process.env.AIRTABLE_BASE_ID)
```

```

1 export function getInvite (inviteCode: string): Promise<Invite> {
2   return new Promise((resolve, reject) => {
3     base('invites')
4       .select({
5         filterByFormula: `{invite} = ${escape(inviteCode)}`, // <- we'll talk more about escape
6         maxRecords: 1
7       })
8     .firstPage((err, records) => {
9       if (err) {
10        console.error(err)
11        return reject(err)
12      }
13
14      if (!records || records.length === 0) {
15        return reject(new Error('Invite not found'))
16      }
17
18      resolve({
19        code: String(records[0].fields.invite),
20        name: String(records[0].fields.name),
21        favouriteColor: String(records[0].fields.favouriteColor),
22        weapon: String(records[0].fields.weapon),
23        coming: typeof records[0].fields.coming === 'undefined'
24          ? undefined
25          : records[0].fields.coming === 'yes'
26      })
27    })
28  })
29 }

```


STEP 3.

NEXT.JS INVITE API

APIS WITH NEXT.JS

Files inside *pages/api* are API endpoints

```
1 // pages/api/hello.ts -> <host>/api/hello
2
3 import type { NextApiResponse, NextApiRequest } from 'next'
4
5 export default async function handler (
6   req: NextApiRequest,
7   res: NextApiResponse<{ message: string }>
8 ) {
9   return res.status(200).json({ message: 'Hello World' })
10 }
```

```

1 // pages/api/invite.ts
2 import { InviteResponse } from '../types/invite'
3 import { getInvite } from '../utils/airtable'
4
5 export default async function handler (
6   req: NextApiRequest,
7   res: NextApiResponse<InviteResponse | { error: string }>
8 ) {
9   if (req.method !== 'GET') {
10     return res.status(405).json({ error: 'Method Not Allowed' })
11   }
12   if (!req.query.code) {
13     return res.status(400).json({ error: 'Missing invite code' })
14   }
15
16   const code = Array.isArray(req.query.code) ? req.query.code[0] : req.query.code
17
18   try {
19     const invite = await getInvite(code)
20     res.status(200).json({ invite })
21   } catch (err) {
22     if ((err as Error).message === 'Invite not found') {
23       return res.status(401).json({ error: 'Invite not found' })
24     }
25     res.status(500).json({ error: 'Internal server error' })
26   }
27 }

```

TESTING

```
curl -XGET "http://localhost:3000/api/invite?code=14b25700-fe5b-45e8-a9be-4863b6239fcf"
```



```
{  
  "invite": {  
    "code": "14b25700-fe5b-45e8-a9be-4863b6239fcf",  
    "name": "Leonardo",  
    "favouriteColor": "blue",  
    "weapon": "Twin Katana"  
  }  
}
```

STEP 4.

INVITE VALIDATION IN REACT

ATTACK PLAN



- When the SPA loads:
 - We grab the invite code from the URL
 - We call the invite API with the code
 -  If it's valid, we render the content
 -  If it's invalid, we render an error

HOW DO WE MANAGE THIS DATA FETCHING LIFECYCLE? 🥲

- In-line in the top-level component (App)?
- In a **Context** provider?
- In a specialized **React Hook**? ☐

HOW CAN WE CREATE A CUSTOM REACT HOOK? 🧐

- A custom Hook is a JavaScript function whose name starts with "use" and that may call other Hooks
- It doesn't need to have a specific signature
- Inside the function, all the common rules of hooks apply:
 - Only call Hooks at the top level
 - Don't call Hooks inside loops, conditions, or nested functions
- reactjs.org/docs/hooks-custom.html


```

1 // components/hooks/useInvite.tsx
2 import { useState, useEffect } from 'react'
3 import { InviteResponse } from '../../types/invite'
4 async function fetchInvite (code: string): Promise<InviteResponse> {
5   // makes a fetch request to the invite api (elided for brevity)
6 }
7
8 export default function useInvite (): [InviteResponse | null, string | null] {
9   const [inviteResponse, setInviteResponse] = useState<InviteResponse | null>(null)
10  const [error, setError] = useState<string | null>(null)
11
12  useEffect(() => {
13    const url = new URL(window.location.toString())
14    const code = url.searchParams.get('code')
15
16    if (!code) {
17      setError('No code provided')
18    } else {
19      fetchInvite(code)
20        .then(setInviteResponse)
21        .catch(err => {
22          setError(err.message)
23        })
24    }
25  }, [])
26
27  return [inviteResponse, error]
28 }

```

EXAMPLE USAGE:

```
1 import React from 'react'
2 import useInvite from './hooks/useInvite'
3
4 export default function SomeExampleComponent () {
5   const [inviteResponse, error] = useInvite()
6
7   // there was an error
8   if (error) {
9     return <div>... some error happened</div>
10  }
11
12  // still loading the data from the backend
13  if (!inviteResponse) {
14    return <div>Loading ...</div>
15  }
16
17  // has the data!
18  return <div>
19    actual component markup when inviteResponse is available
20  </div>
21 }
```

STEP 5.

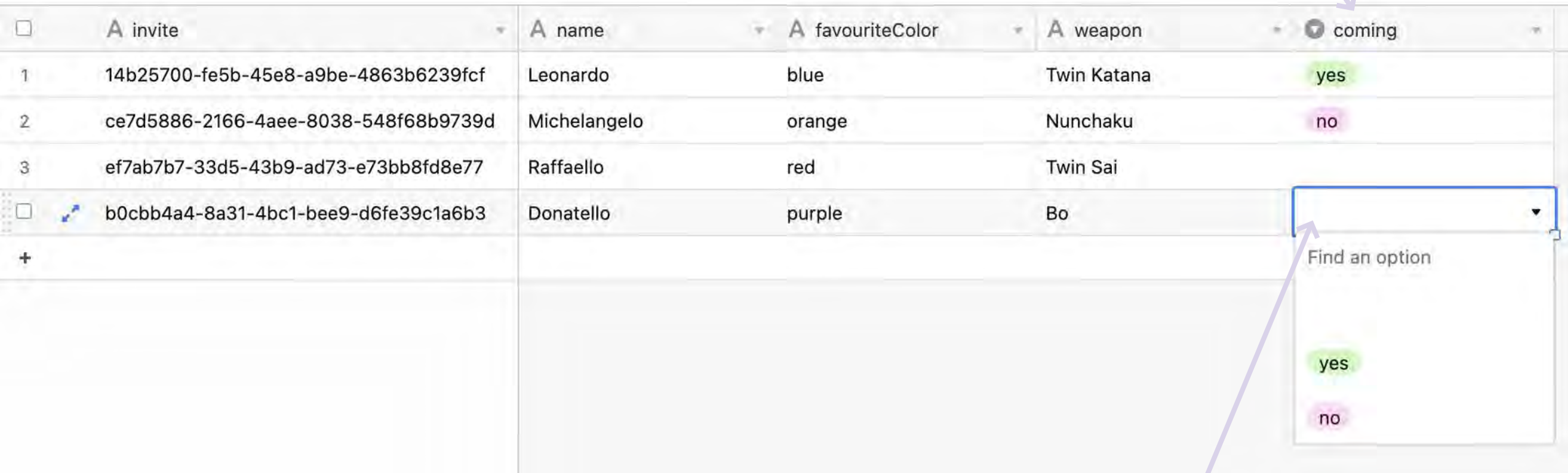
COLLECTING USER DATA


CHANGES REQUIRED 🙄

- Add the "coming" field in Airtable
- Add a new backend utility to update the "coming" field for a given invite
- Add a new endpoint to update the coming field for the current user
- Update the React hook to expose the update functionality

ADD THE 'COMING' FIELD IN AIRTABLE

New field



	A invite	A name	A favouriteColor	A weapon	coming
1	14b25700-fe5b-45e8-a9be-4863b6239fcf	Leonardo	blue	Twin Katana	yes
2	ce7d5886-2166-4aee-8038-548f68b9739d	Michelangelo	orange	Nunchaku	no
3	ef7ab7b7-33d5-43b9-ad73-e73bb8fd8e77	Raffaello	red	Twin Sai	
	 b0cbb4a4-8a31-4bc1-bee9-d6fe39c1a6b3	Donatello	purple	Bo	<div data-bbox="2091 636 2519 1053"><p>Find an option</p><p>yes</p><p>no</p></div>
+					

("yes", "no", or undefined)

RSVP UTILITY

```
1 // utils/airtable.ts
2 import Airtable, { FieldSet, Record } from 'airtable'
3 // ...
4
5 export function getInviteRecord (inviteCode: string): Promise<Record<FieldSet>> {
6   // gets the raw record for a given invite, elided for brevity
7 }
8
9 export async function updateRsvp (inviteCode: string, rsvp: boolean): Promise<void> {
10   const { id } = await getInviteRecord(inviteCode)
11
12   return new Promise((resolve, reject) => {
13     base('invites').update(id, { coming: rsvp ? 'yes' : 'no' }, (err) => {
14       if (err) {
15         return reject(err)
16       }
17
18       resolve()
19     })
20   })
21 }
```

RSVP API ENDPOINT

```
1 // pages/api/rsvp.ts
2 type RequestBody = {coming?: boolean}
3
4 export default async function handler (
5   req: NextApiRequest,
6   res: NextApiResponse<{updated: boolean} | { error: string }>
7 ) {
8   if (req.method !== 'PUT') {
9     return res.status(405).json({ error: 'Method Not Allowed' })
10  }
11  if (!req.query.code) {
12    return res.status(400).json({ error: 'Missing invite code' })
13  }
14  const reqBody = req.body as RequestBody
15  if (typeof reqBody.coming === 'undefined') {
16    return res.status(400).json({ error: 'Missing `coming` field in body' })
17  }
18  const code = Array.isArray(req.query.code) ? req.query.code[0] : req.query.code
19
20  try {
21    await updateRsvp(code, reqBody.coming)
22    return res.status(200).json({ updated: true })
23  } catch (err) {
24    if ((err as Error).message === 'Invite not found') {
25      return res.status(401).json({ error: 'Invite not found' })
26    }
27    res.status(500).json({ error: 'Internal server error' })
28  }
29 }
```

USEINVITE HOOK V2

```
1 // components/hooks/useInvite.tsx
2 // ...
3
4 interface HookResult {
5   inviteResponse: InviteResponse | null,
6   error: string | null,
7   updating: boolean,
8   updateRsvp: (coming: boolean) => Promise<void>
9 }
10
11 async function updateRsvpRequest (code: string, coming: boolean): Promise<void> {
12   // Helper function that uses fetch to invoke the rsvp API endpoint (elided)
13 }
```



```
1 // ...
2 export default function useInvite (): HookResult {
3     const [inviteResponse, setInviteResponse] = useState<InviteResponse | null>(null)
4     const [error, setError] = useState<string | null>(null)
5     const [updating, setUpdating] = useState<boolean>(false)
6
7     useEffect(() => {
8         // load the invite using the code from URL, same as before
9     }, [])
10
11     async function updateRsvp (coming: boolean) {
12         if (inviteResponse) {
13             setUpdating(true)
14             await updateRsvpRequest(inviteResponse.invite.code, coming)
15             setInviteResponse({
16                 ...inviteResponse,
17                 invite: { ...inviteResponse.invite, coming }
18             })
19             setUpdating(false)
20         }
21     }
22
23     return { inviteResponse, error, updating, updateRsvp }
24 }
```

USING THE HOOK

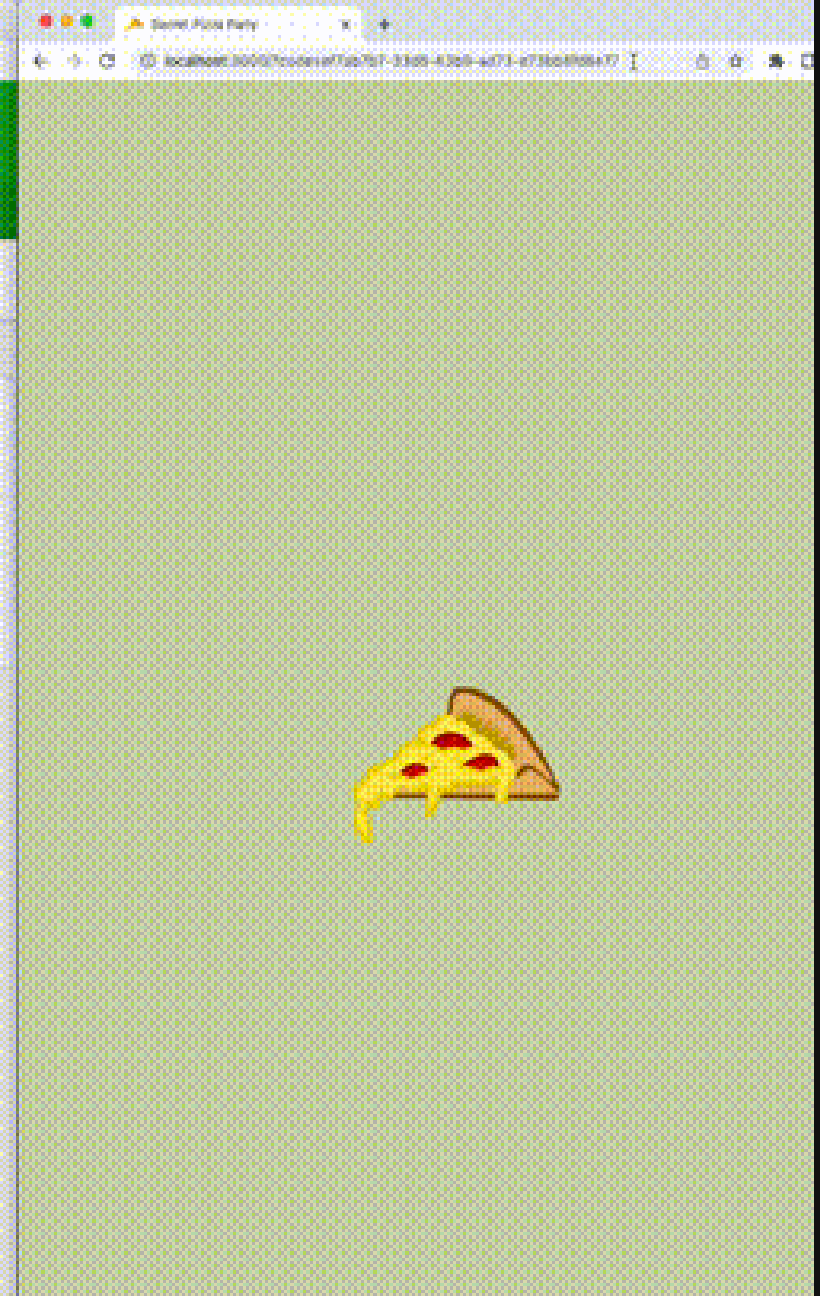
```
1 import useInvite from './hooks/useInvite'
2
3 export default function Home () {
4   const { inviteResponse, error, updating, updateRsvp } = useInvite()
5
6   if (error) { return <div>Duh! {error}</div> }
7   if (!inviteResponse) { return <div>Loading...</div> }
8
9   function onRsvpChange (e: ChangeEvent<HTMLInputElement>) {
10     const coming = e.target.value === 'yes'
11     updateRsvp(coming)
12   }
13
14   return (<fieldset disabled={updating}><legend>Are you coming?</legend>
15     <label htmlFor="yes">
16       <input type="radio" id="yes" name="coming" value="yes"
17         onChange={onRsvpChange}
18         checked={inviteResponse.invite.coming === true}
19       /> YES
20     </label>
21     <label htmlFor="no">
22       <input type="radio" id="no" name="coming" value="no"
23         onChange={onRsvpChange}
24         checked={inviteResponse.invite.coming === false}
25       /> NO
26     </label>
27   </fieldset>)
28 }
```



15 SECONDS DEMO 

	A invite	A weapon	coming
1	14b25700-fe5b-45e8-a9be-4863b6239fcd	Twin Katana	
2	ce7d5886-2166-4aee-8038-548f68b9739d	Nunchaku	
3	ef7ab7b7-33d5-43b9-ad73-e73bb8fd8e77	Twin Sai	
4	b0cbb4a4-8a31-4bc1-bee9-d6fe39c1a6b3	Bo	
+			

Create...
Grid +
Form +
Calendar +
Gallery +
Kanban +
Time... Pro +



DEPLOYMENT

secret-pizza-party

[View Git Repository](#) [Visit](#)

Production Deployment

[View Build Logs](#) [View Function Logs](#) [View Domains](#)

The deployment that is available to your visitors.



DEPLOYMENT
secret-pizza-party-4qg1agchx-lmammino.vercel.app

DOMAINS
secret-pizza-party.vercel.app +2

STATUS CREATED
● Ready 1d ago

BRANCH
main
Added code to handle state update.

To update your Production Deployment, push to the "main" branch.

[Learn More](#)

Preview Deployments

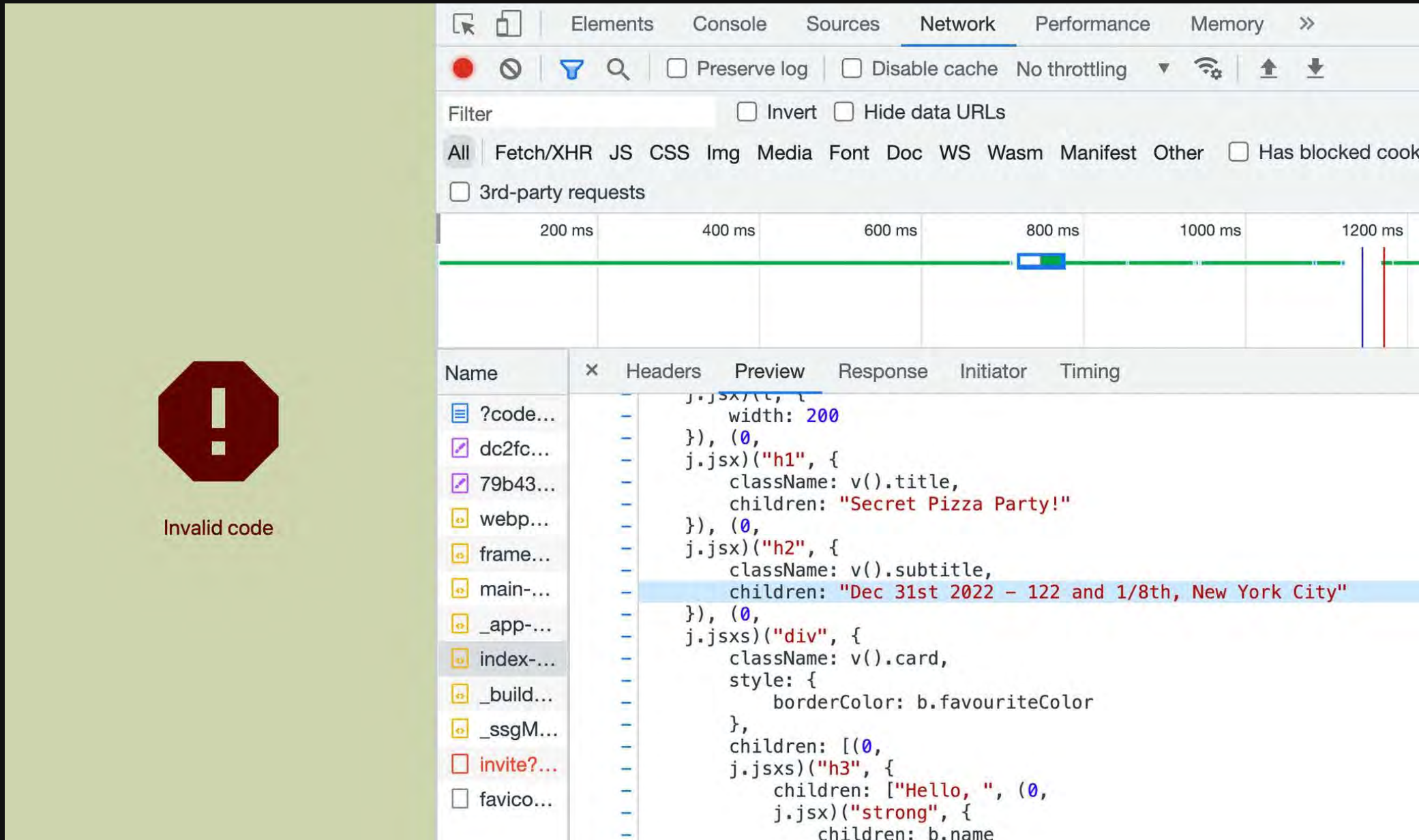
Status 4/5

secret-pizza-party-ik6oe21zt-lmammino.verc... Preview	● Ready 26s	made it vulnerable by removing escape functio... vulnerable	3d ago by lmmammino
secret-pizza-party-fgpypfb66-lmammino.verc... Preview	● Ready 26s	made it vulnerable by removing escape functio... vulnerable	3d ago by lmmammino
secret-pizza-party-lo617tzfx-lmammino.verc... Preview	● Ready 26s	made it vulnerable by removing escape functio... vulnerable	3d ago by lmmammino

SECURITY CONSIDERATIONS

What if I don't have an invite code and I want to *hack* into the website anyway?

DISCLOSING SENSITIVE INFORMATION IN THE SOURCE CODE



The screenshot shows a web browser interface. On the left, a red octagonal error icon with a white exclamation mark is displayed above the text "Invalid code". The main content area is a light green background. On the right, the browser's developer tools are open to the Network tab. The top toolbar shows various icons for navigation and filtering. The Network tab shows a list of requests, with the selected request expanded to show its response. The response is a JSON object containing sensitive information, including a date and location: "Dec 31st 2022 - 122 and 1/8th, New York City".

Invalid code

Network tab response:

```
width: 200
}), (0,
j.jsx("h1", {
  className: v().title,
  children: "Secret Pizza Party!"
}), (0,
j.jsx("h2", {
  className: v().subtitle,
  children: "Dec 31st 2022 - 122 and 1/8th, New York City"
}), (0,
j.jsx("div", {
  className: v().card,
  style: {
    borderColor: b.favouriteColor
  },
  children: [(0,
j.jsx("h3", {
  children: ["Hello, ", (0,
j.jsx("strong", {
  children: b.name
```


HOW DO WE FIX THIS?

- Don't hardcode any sensitive info in your JSX (or JS in general)
- Use the invite API to return any sensitive info (together with the user data)
- This way, the sensitive data is available only in the backend code

AIRTABLE FILTER FORMULA INJECTION

```
1 export function getInvite (inviteCode: string): Promise<Invite> {
2   return new Promise((resolve, reject) => {
3     base('invites')
4       .select({
5         filterByFormula: `{invite} = ${escape(inviteCode)}`,
6         maxRecords: 1
7       })
8     .firstPage((err, records) => {
9       // ...
10    })
11  })
12 }
```

AIRTABLE FILTER FORMULA INJECTION

```
1 export function getInvite (inviteCode: string): Promise<Invite> {
2   return new Promise((resolve, reject) => {
3     base('invites')
4     .select({
5       filterByFormula: `{invite} = '${inviteCode}'`,
6       maxRecords: 1
7     })
8     .firstPage((err, records) => {
9       // ...
10    })
11  })
12 }
```

inviteCode is user controlled!

The user can change this value arbitrarily! 🤖

SO WHAT?!

If the user inputs the following query string:

```
?code=14b25700-fe5b-45e8-a9be-4863b6239fcf
```

We get the following filter formula

```
{invite} = '14b25700-fe5b-45e8-a9be-4863b6239fcf'
```



SO WHAT?!

But, if the user inputs this other query string: 😈

```
?code=%27%20>%3D%200%20%26%20%27
```

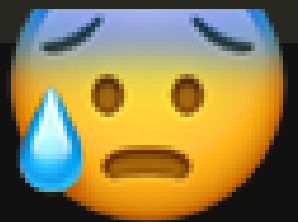
Which is basically the following unencoded query string:

```
?code=' >= 0 & '
```

Now we get:

```
{invite} = '' >= 0 & ''
```

which is TRUE for EVERY RECORD!





Secret Pizza Party!

Dec 31st 2022 - 122 and 1/8th, New York City

Hello, **Donatello!**

You have been invited to the most awesome secret pizza party of the year!

Are you coming?

- Cowabunga! (yes)
- Nitwits! (no)

P.S. You don't need to bring your **Bo**, *Shredder* is not invited!



HOW DO WE FIX THIS?

- The escape function "sanitizes" user input to try to prevent injection
- Unfortunately Airtable does not provide an official solution for this, so this escape function is the best I could come up with, but it might not be "good enough"! 🙄

```
function escape (value: string): string {  
  if (value === null ||  
      typeof value === 'undefined') {  
    return 'BLANK()'  
  }  
  
  if (typeof value === 'string') {  
    const escapedString = value  
      .replace(/'/g, "\\'  
      .replace(/\\r/g, '\\r'  
      .replace(/\\/g, '\\\\'  
      .replace(/\\n/g, '\\n'  
      .replace(/\\t/g, '\\t'  
    return `\\${escapedString}`  
  }  
  
  if (typeof value === 'number') {  
    return String(value)  
  }  
  
  if (typeof value === 'boolean') {  
    return value ? '1' : '0'  
  }  
  
  throw Error('Invalid value received')  
}
```

LET'S WRAP THINGS UP...



LIMITATIONS

Airtable API rate limiting: 5 req/sec 🥵

(We actually do 2 calls when we update a record!)

POSSIBLE ALTERNATIVES 🙄

Google spreadsheet (there's an [API](#) and a [package](#))

DynamoDB (with Amplify)

Firebase (?)

Any headless CMS (?)

Supabase or Strapi (?)

TAKEAWAYS

- This solution is a **quick, easy, and cheap** way to build invite-only websites.
- We learned about **Next.js API endpoints, custom React Hooks,** how to use **AirTable** (and its SDK), and a bunch of **security** related things.
- Don't use this solution blindly: **evaluate your context and find the best tech stack!**

ALSO AVAILABLE AS AN ARTICLE

With the full [codebase on GitHub!](#)



loige.link/microsite-article



THANKS! 🙌🙌

loige.link/micro42



fourtheorem.com

Photo by [Grant Durr](#) on [Unsplash](#)