ChatGPT

Next.js Admin Screen for Tax Obligations

This solution provides a **production-grade Next.js 13** admin screen for the Tax Intelligence Extension (Brazil). It uses **Tailwind CSS** and **ShadCN UI** components to create a clean, minimal interface. The screen displays a grid of tax obligations with filters and includes a detail drawer for audit trails. All components are modular, reusable, and use dummy data with comments indicating where to insert dynamic data.

UI Features and Structure

- **Responsive Obligations Grid:** Uses ShadCN's `<Table>` components to list tax obligations with columns for Entity Name, Period, Obligation Code, Tax Base, Tax Due, and Status [1]. Status labels are styled with ShadCN `<Badge>` variants (e.g. grey for pending, green for submitted, red for overdue).
- **Filter Controls:** Two dropdown filters (Period and Obligation Code) allow refining the list. These are built with ShadCN's `<Select>` component for a consistent look. When a filter is selected, the table data updates to show matching records.
- **Audit Trail Drawer:** Each obligation row has a **View** button. Clicking it opens a side drawer (using ShadCN's `<Drawer>` component) showing the audit trail for that obligation [2]. The drawer displays a list of events with dates, actions, linked evidence IDs, and output file names (dummy links for now).
- **Modular Layout:** The code is split into a Next.js App Router layout (providing a persistent header, main content container, and footer) and the main page component. This ensures a minimal layout structure with a header at the top, the data grid in the main section, and a footer at the bottom.

Implementation Details

We use Next.js 13 App Router file conventions. The `layout.tsx` defines the overall page structure (header, main, footer), and the `page.tsx` implements the admin screen's content. The page component is a **Client Component** (`"use client"`) since it manages state (filters and drawer toggles) on the client side. All UI elements use **ShadCN UI** components (imported from `@/components/ui/...`) for consistency. Dummy data is provided for obligations and audit trails, and comments in the code indicate where to replace with real data from your backend or API.

Below is the complete code. You can copy these into a Next.js 13+ project (for example, under `app/admin/`) and they will run with minimal setup (assuming Tailwind and ShadCN UI are configured in the project):

`app/admin/layout.tsx` – Layout with Header and Footer

This layout component defines a simple header, a main container for content, and a footer. It uses Tailwind utility classes for spacing and borders. The `{children}` will render the page content (the obligations table and drawer).

```tsx
import React, { ReactNode } from "react";

export const metadata = {
  title: "Tax Obligations Admin",
};

interface AdminLayoutProps {
  children: ReactNode;
}

export default function AdminLayout({ children }: AdminLayoutProps) {
  return (
    <div className="flex min-h-screen flex-col">
      {/* Header */}
      <header className="border-b bg-white p-4">
        <h1 className="text-lg font-bold">Tax Intelligence Admin</h1>
      </header>

      {/* Main content area */}
      <main className="flex-1">
        <div className="max-w-7xl mx-auto p-4">{children}</div>
      </main>

      {/* Footer */}
      <footer className="border-t bg-white p-4 text-center text-sm text-muted-
foreground">
        {/* Footer text can include company or system name */}
        © 2025 Tax Intelligence Extension – Brazil
      </footer>
    </div>
  );
}
```

`app/admin/page.tsx` – Admin Page with Table, Filters, and Drawer

This is the main page component displaying the tax obligations table with filters and the audit trail drawer. It uses dummy data for now. The code includes comments indicating where to fetch or pass in real data and how to handle actions. The page is marked as a client component to enable stateful filtering and interactive drawers.

```tsx
"use client";

import React, { useState } from "react";
import {
  Table, TableBody, TableCaption, TableCell, TableHead, TableHeader, TableRow,
```

```
TableFooter,
} from "@/components/ui/table";
import { Badge } from "@/components/ui/badge";
import {
  Select, SelectTrigger, SelectContent, SelectItem, SelectValue,
} from "@/components/ui/select";
import { Button } from "@/components/ui/button";
import {
  Drawer, DrawerTrigger, DrawerContent, DrawerHeader, DrawerTitle,
  DrawerDescription, DrawerFooter, DrawerClose,
} from "@/components/ui/drawer";

// Define TypeScript types for obligations and audit events (for clarity and
type-checking)
interface AuditEvent {
  date: string;
  action: string;
  evidenceId?: string;
  outputFile?: string;
}
interface TaxObligation {
  id: number;
  entityName: string;
  period: string;
  obligationCode: string;
  taxBase: string;
  taxDue: string;
  status: string;
  auditTrail: AuditEvent[];
}

// Dummy data for tax obligations – replace with real data from API or props
const obligations: TaxObligation[] = [
  {
    id: 1,
    entityName: "ABC Corp",
    period: "2023-01",
    obligationCode: "OB001",
    taxBase: "R$ 10,000",
    taxDue: "R$ 1,500",
    status: "Pending",
    auditTrail: [
      {
        date: "2024-01-15",
        action: "Obligation Created",
        evidenceId: "EV-1001",
        outputFile: undefined,
      },
```

```
      {
        date: "2024-02-10",
        action: "Submitted to Tax Authority",
        evidenceId: "EV-1002",
        outputFile: "submission_receipt.pdf",
      },
    ],
  },
  {
    id: 2,
    entityName: "XYZ Ltd",
    period: "2023-01",
    obligationCode: "OB002",
    taxBase: "R$ 8,500",
    taxDue: "R$ 1,275",
    status: "Submitted",
    auditTrail: [
      {
        date: "2024-02-05",
        action: "Obligation Created",
        evidenceId: "EV-1100",
        outputFile: undefined,
      },
      {
        date: "2024-02-20",
        action: "Submitted to Tax Authority",
        evidenceId: "EV-1101",
        outputFile: "submission_receipt.pdf",
      },
      {
        date: "2024-03-01",
        action: "Confirmation Received",
        evidenceId: "EV-1102",
        outputFile: "confirmation_notice.pdf",
      },
    ],
  },
  {
    id: 3,
    entityName: "ACME Inc",
    period: "2023-02",
    obligationCode: "OB001",
    taxBase: "R$ 12,000",
    taxDue: "R$ 1,800",
    status: "Overdue",
    auditTrail: [
      {
        date: "2024-03-10",
```

```
        action: "Obligation Created",
        evidenceId: "EV-1200",
        outputFile: undefined,
      },
      {
        date: "2024-04-15",
        action: "Submission Late - Penalty Applied",
        evidenceId: "EV-1201",
        outputFile: "late_submission.pdf",
      },
    ],
  },
  // ... add more dummy records as needed
];

export default function TaxObligationsPage() {
  // State for filters (null/empty string means no filter applied)
  const [periodFilter, setPeriodFilter] = useState<string>("");
  const [codeFilter, setCodeFilter] = useState<string>("");

  // Compute unique period and obligation code options from data for filter
dropdowns
  const periods = Array.from(new Set(obligations.map((ob) =>
ob.period))).sort();
  const codes = Array.from(new Set(obligations.map((ob) =>
ob.obligationCode))).sort();

  // Filtered obligations based on selected filters
  const filteredObligations = obligations.filter((ob) => {
    const matchesPeriod = periodFilter ? ob.period === periodFilter : true;
    const matchesCode = codeFilter ? ob.obligationCode === codeFilter : true;
    return matchesPeriod && matchesCode;
  });

  // Optional: mapping of status to badge color variants (using ShadCN Badge
variants)
  const statusVariant: Record<string, "default" | "secondary" | "destructive"
| "outline"> = {
    "Pending": "secondary",    // gray badge
    "Submitted": "default",    // default style badge (e.g., primary accent)
    "Overdue": "destructive",  // red badge
  };

  return (
    <div className="space-y-6">  {/* Container for page content with vertical
spacing */}
      {/* Page title (could also be in header, but included here for clarity)
*/}
```

```
<h2 className="text-xl font-semibold">Tax Obligations</h2>

{/* Filters Section */}
<div className="flex flex-wrap items-center gap-4">
  {/* Period Filter Dropdown */}
  <Select
    value={periodFilter}
    onValueChange={(val) => setPeriodFilter(val)}
  >
    <SelectTrigger className="w-[150px]">
      <SelectValue placeholder="All Periods" />
    </SelectTrigger>
    <SelectContent>
      <SelectItem value="">All Periods</SelectItem>
      {periods.map((period) => (
        <SelectItem key={period} value={period}>
          {period}
        </SelectItem>
      ))}
    </SelectContent>
  </Select>

  {/* Obligation Code Filter Dropdown */}
  <Select
    value={codeFilter}
    onValueChange={(val) => setCodeFilter(val)}
  >
    <SelectTrigger className="w-[180px]">
      <SelectValue placeholder="All Obligation Codes" />
    </SelectTrigger>
    <SelectContent>
      <SelectItem value="">All Codes</SelectItem>
      {codes.map((code) => (
        <SelectItem key={code} value={code}>
          {code}
        </SelectItem>
      ))}
    </SelectContent>
  </Select>
</div>

{/* Obligations Table */}
<Table>
  <TableCaption>List of tax obligations and their status</TableCaption>
  <TableHeader>
    <TableRow>
      <TableHead className="w-[200px]">Entity Name</TableHead>
      <TableHead>Period</TableHead>
```

```
            <TableHead>Obligation Code</TableHead>
            <TableHead className="text-right">Tax Base</TableHead>
            <TableHead className="text-right">Tax Due</TableHead>
            <TableHead>Status</TableHead>
            <TableHead className="text-center">Actions</TableHead>
          </TableRow>
        </TableHeader>
        <TableBody>
          {filteredObligations.map((obligation) => (
            <TableRow key={obligation.id}>
              <TableCell className="font-medium">{obligation.entityName}</
TableCell>
              <TableCell>{obligation.period}</TableCell>
              <TableCell>{obligation.obligationCode}</TableCell>
              <TableCell className="text-right">{obligation.taxBase}</TableCell>
              <TableCell className="text-right">{obligation.taxDue}</TableCell>
              <TableCell>
                {/* Status badge (color-coded via variant) */}
                <Badge variant={statusVariant[obligation.status] ?? "outline"}>
                  {obligation.status}
                </Badge>
              </TableCell>
              <TableCell className="text-center">
                {/* Drawer trigger button for audit trail */}
                <Drawer>
                  <DrawerTrigger asChild>
                    <Button variant="outline" size="sm">View</Button>
                  </DrawerTrigger>
                  <DrawerContent className="w-full sm:max-w-lg">
                    <DrawerHeader>
                      <DrawerTitle>Audit Trail</DrawerTitle>
                      <DrawerDescription>
                        {obligation.obligationCode} – {obligation.entityName},
Period {obligation.period}
                      </DrawerDescription>
                    </DrawerHeader>
                    {/* Audit trail events list */}
                    <div className="mt-4 space-y-2">
                      {obligation.auditTrail.map((event, index) => (
                        <div
                          key={index}
                          className="rounded-lg border p-3"
                        >
                          <p className="text-sm"><strong>Date:</strong>
{event.date}</p>
                          <p className="text-sm"><strong>Action:</strong>
{event.action}</p>
                          <p className="text-sm">
```

```
                          <strong>Evidence ID:</strong>{" "}
                          {event.evidenceId ? (
                            <a href="#" className="underline text-blue-600
hover:text-blue-800">
                              {event.evidenceId}
                            </a>
                          ) : "–"}
                        </p>
                        <p className="text-sm">
                          <strong>Output File:</strong>{" "}
                          {event.outputFile ? (
                            <a href="#" className="underline text-blue-600
hover:text-blue-800">
                              {event.outputFile}
                            </a>
                          ) : "–"}
                        </p>
                      </div>
                    ))}
                  </div>
                  <DrawerFooter className="mt-6">
                    {/* Close button to dismiss drawer */}
                    <DrawerClose asChild>
                      <Button variant="outline">Close</Button>
                    </DrawerClose>
                  </DrawerFooter>
                </DrawerContent>
              </Drawer>
            </TableCell>
          </TableRow>
        ))}
        {filteredObligations.length === 0 && (
          <TableRow>
            <TableCell colSpan={7} className="text-center py-6 text-sm text-
muted-foreground">
              No obligations found for the selected filters.
            </TableCell>
          </TableRow>
        )}
      </TableBody>
      {/* (Optional) TableFooter could go here if needed for totals or legend
*/}
    </Table>
  </div>
 );
}
```

Notes: - The `<Select>` placeholders "All Periods" and "All Codes" act as a no-filter state (value set to an empty string). Selecting these resets the corresponding filter. In a real app, you might populate filter options from an API or context. - The table uses ShadCN's UI components for a consistent design [1]. We apply Tailwind classes (e.g. `text-right`, `w-[200px]`) to adjust alignment and width for certain columns as needed. - Each table row contains its own `<Drawer>` component. The **View** button is wrapped in a `DrawerTrigger` to open the drawer, and the `DrawerContent` displays the audit trail details for that specific obligation [2]. This approach keeps the drawer content tied to the relevant row's data. Only one drawer will open at a time (when its trigger is clicked). The drawer can be closed by clicking outside or using the provided **Close** button. - The audit trail is presented as a list of events. In this dummy setup, each event shows Date, Action, Evidence ID, and Output File. Evidence IDs and Output files are rendered as hyperlinks (`<a href="#">`) for demonstration. In production, you would set these to actual URLs or actions that open the evidence documents or files. - All data is currently static. In a real application, you would fetch this data from your database or API (e.g., using Next.js data fetching in a server component, or via an API route). For example, you could replace the `obligations` array with data fetched in a server component and passed to this client component as props. The code is structured so that integrating real data should be straightforward (e.g., mapping over fetched obligations in place of the dummy array).

Conclusion

The above code provides a ready-to-use Next.js 13 page for managing tax obligations. It demonstrates a clean UI with Tailwind and ShadCN UI, and a modular structure separating layout and page logic. **To use this in your project**, ensure you have ShadCN UI components installed for Button, Table, Select, Drawer, etc., and Tailwind CSS configured. Then copy the `layout.tsx` and `page.tsx` into your app (adjusting the path/naming as needed). Finally, replace the mock data and placeholders with real data from the Tax Intelligence Extension system. This will give you a functional admin dashboard screen with filtering and detail views, ready for further integration and customization. [3] [4]

---

[1] [3] Table - shadcn/ui
https://ui.shadcn.com/docs/components/table

[2] [4] Drawer - shadcn/ui
https://ui.shadcn.com/docs/components/drawer