

# Kapitel 22. Analyse des Architekturrisikos

---

Diese Arbeit wurde mithilfe von KI übersetzt. Wir freuen uns über dein Feedback und deine Kommentare: [translation-feedback@oreilly.com](mailto:translation-feedback@oreilly.com)

---

Jede Architektur birgt Risiken: einige operative (wie Verfügbarkeit, Skalierbarkeit und Datenintegrität), andere strukturelle (wie die statische Kopplung zwischen logischen Komponenten). Die Analyse des Architekturrisikos ist eine der wichtigsten Aufgaben von Architekten. Sie ermöglicht es ihnen, Schwachstellen und strukturelle Verfallserscheinungen in der Architektur zu erkennen und Korrekturmaßnahmen zu ergreifen. In diesem Kapitel zeigen wir dir einige Schlüsseltechniken und -praktiken für die Quantifizierung, Bewertung und Identifizierung von Risiken und stellen eine Aktivität vor, die als *Risk Storming* bezeichnet wird.

## Risiko-Matrix

Bei der Bewertung des Architekturrisikos muss zunächst der Grad des Risikos bestimmt werden: ob das Risiko für einen bestimmten Teil der Architektur gering, mittel oder hoch ist. Die Herausforderung dabei ist, dass die Bewertung des Risikos *subjektiv* sein kann. Ein Architekt könnte der *Meinung* sein, dass ein bestimmter Aspekt der Architektur ein hohes Risiko darstellt, während ein anderer Architekt denselben Aspekt für ein

mittleres Risiko *hält*. Wir betonen die Subjektivität der Risikobewertung, indem wir die *Meinung* hier kursiv setzen. Glücklicherweise haben Architekten eine nützliche Risikobewertungsmatrix, die uns hilft, Risiken messbar zu machen.

Die Risikobewertungsmatrix für die Architektur([Abbildung 22-1](#)) verwendet zwei Dimensionen, um das Risiko zu bewerten: die Gesamtauswirkung des betreffenden Risikos und die Wahrscheinlichkeit, dass das Risiko eintritt. Der Architekt stuft jede Dimension als niedrig (1), mittel (2) oder hoch (3) ein und multipliziert dann die Zahlen in jedem Schnittpunkt der Matrix. Auf diese Weise erhält man eine numerische Darstellung des Risikos, die den Prozess der Risikoeinstufung *objektiver* macht. Die Zahlen 1 und 2 gelten als geringes Risiko (normalerweise grün), die Zahlen 3 und 4 als mittleres Risiko (normalerweise gelb) und die Zahlen 6 bis 9 als hohes Risiko (normalerweise rot). Die Verwendung von Schattierungen kann bei der Darstellung in Graustufen und für Menschen, die keine Farben unterscheiden können, hilfreich sein.

		Likelihood of risk occurring		
		Low (1)	Medium (2)	High (3)
Overall impact of risk	Low (1)	1	2	3
	Medium (2)	2	4	6
	High (3)	3	6	9

Abbildung 22-1. Matrix zur Bestimmung des Architekturrisikos

Um dir ihren Nutzen zu verdeutlichen, werden wir die Risikomatrix in einem Beispiel verwenden. Angenommen, du machst dir Sorgen um die Verfügbarkeit der wichtigsten zentralen Datenbank deiner Anwendung.

---

#### TIPP

Wenn du diese Matrix zur Risikoeinstufung verwendest, solltest du zuerst die Auswirkungen und dann die Wahrscheinlichkeit berücksichtigen. Wenn du dir unsicher bist, wie hoch die Wahrscheinlichkeit ist, nimm eine hohe Bewertung (3), bis du sie bestätigen kannst.

---

Betrachte zunächst die Gesamtauswirkungen: Was passiert, wenn die Datenbank ausfällt oder nicht mehr verfügbar ist? Angenommen, du stufst die Auswirkungen als hohes Risiko ein und ordnest dieses Risiko in der letzten Zeile der Matrix in [Abbildung 22-1](#) als 3 (mittel), 6 (hoch) oder 9 (hoch) ein. Wenn du jedoch die zweite Dimension - die Wahrscheinlichkeit des Eintretens dieses Risikos - betrachtest, stellst du fest, dass die Datenbank auf hochverfügbaren Servern in einer Clusterkonfiguration liegt und die *Wahrscheinlichkeit*, dass die Datenbank nicht mehr verfügbar ist, gering ist (erste Spalte in der Matrix). Die Schnittmenge aus hoher Auswirkung und geringer Wahrscheinlichkeit ergibt eine Gesamtrisikobewertung von 3 (mittleres Risiko) für die Verfügbarkeit der primären zentralen Datenbank.

## Risikobewertungen

Du kannst die Risikomatrix verwenden, um eine sogenannte *Risikobewertung* zu erstellen: einen zusammenfassenden Bericht über das Gesamtrisiko einer Architektur mit aussagekräftigen Bewertungskriterien, die auf einem Kontext basieren (Dienste, Subdomänenbereiche oder Domänenbereiche eines Systems). Wir haben schon viele Risikobewertungen durchgeführt und festgestellt,

dass sich Architekturmerkmale hervorragend als Risikobewertungskriterien eignen. Warum sollte man Zeit auf die Analyse des Leistungsrisikos verwenden, wenn die kritischen architektonischen Merkmale des Systems die Skalierbarkeit, Elastizität und Datenintegrität sind? Die Kenntnis von Merkmalen wie den in [Kapitel 4](#) beschriebenen ist der erste Schritt bei der Analyse des Architekturrisikos.

---

#### TIPP

Die Architekturmerkmale, die für die Unterstützung der Architektur am wichtigsten sind, eignen sich hervorragend als Kriterien für die Risikobewertung.

---

Das grundlegende Format eines Risikobewertungsberichts ist in [Abbildung 22-2](#) dargestellt. Hier stehen 1 und 2 für ein geringes Risiko, 3 und 4 für ein mittleres Risiko und 6 und 9 für ein hohes Risiko. Die Risikokriterien befinden sich auf der linken Seite des Arbeitsblatts und der Kontext befindet sich oben.

RISK CRITERIA	Customer registration	Catalog checkout	Order fulfillment	Order shipment	TOTAL RISK
Scalability	2	6	1	2	11
Availability	3	4	2	1	10
Performance	4	2	3	6	15
Security	6	3	1	1	11
Data integrity	9	6	1	1	17
TOTAL RISK	24	21	8	11	

Abbildung 22-2. Beispiel für eine Standard-Risikobewertung

Wir verwenden ein E-Commerce-Bestellsystem als Beispiel für diese Risikobewertung. Es werden fünf Kriterien bewertet, die die kritischen Architekturmerkmale des Systems darstellen. Oben befinden sich vier

verschiedene Kontexte, die jeweils eine eigene Domäne darstellen (Kundenregistrierung, Katalog-Checkout, Bestellabwicklung und Bestellversand). Ein Domänen- oder Subdomänenkontext eignet sich gut; eine Risikoanalyse auf Ebene der Dienste ist in der Regel zu feinkörnig und berücksichtigt nicht die Risiken, die mit der Kommunikation oder Koordination zwischen mehreren Diensten verbunden sind.

Das Schöne an der Verwendung des quantifizierten Risikos ist, dass es sowohl die Risikokriterien als auch den Kontext berücksichtigt. In [Abbildung 22-2](#) beträgt das kumulierte Gesamtrisiko für die Datenintegrität beispielsweise 17 und ist damit aus Sicht der Kriterien der höchste Risikobereich. Das kumulierte Risiko für die Verfügbarkeit beträgt nur 10 (das niedrigste Risiko nach den Kriterien). Hinsichtlich des relativen Risikos der einzelnen Kontextbereiche ist die Kundenregistrierung der Bereich mit dem höchsten Risiko, während die Auftragsabwicklung das geringste Risiko darstellt. Dies ist eine nützliche Information, wenn es darum geht, Prioritäten zu setzen und festzustellen, wo zusätzliche Anstrengungen zur Risikominderung unternommen werden sollten.

Dieses Beispiel für eine Risikobewertung enthält alle Ergebnisse der Risikoanalyse, aber manchmal ist es sinnvoll, Details herauszufiltern, um bestimmte Probleme hervorzuheben. Nimm zum Beispiel an, dass du als Architekt dieses Systems in einer Besprechung bist und den Interessengruppen die risikoreichen Bereiche des Systems vorstellst. Anstatt die gesamte Risikobewertung zu präsentieren, wie in [Abbildung](#)

22-2, könntest du die Bereiche mit geringem und mittlerem Risiko (das Rauschen) herausfiltern, um die Bereiche mit hohem Risiko (das Signal) hervorzuheben. Durch die Verbesserung des Signal-Rausch-Verhältnisses kannst du eine effektivere und weniger ablenkende Botschaft vermitteln. Abbildung 22-3 zeigt eine gefilterte Version der gleichen Risikobewertung. Vergleiche die beiden Bilder, um zu sehen, wie viel klarer die Botschaft in der zweiten gefilterten Bewertung ist.

RISK CRITERIA	Customer registration	Catalog checkout	Order fulfillment	Order shipment	TOTAL RISK
Scalability		6			6
Availability					0
Performance				6	6
Security	6				6
Data integrity	9	6			15
TOTAL RISK	15	12	0	6	

Abbildung 22-3. Filtern der Risikobewertung, um nur hohe Risiken anzuzeigen

Ein Problem mit der vollständigen Version der Risikobewertung ist, dass sie nur eine Momentaufnahme darstellt und nicht zeigt, ob sich die Situation verbessert oder verschlechtert. Mit anderen Worten:

Abbildung 22-2 zeigt nicht die *Richtung des Risikos*. Du kannst die Richtung des Risikos bestimmen, indem du kontinuierliche Messungen mit Hilfe von Fitnessfunktionen durchführst, wie in Kapitel 6 beschrieben. Wenn du jedes Risikokriterium objektiv analysierst, kannst du Trends beobachten, um die Richtung der einzelnen Risikokriterien zu erkennen.

In Abbildung 22-4 fügen wir der Risikobewertung eine dritte Dimension hinzu: die *Richtung*. Ein Dreieck, das auf dem Kopf steht, bedeutet, dass sich das Risiko für ein bestimmtes Kriterium und einen bestimmten Kontext verschlechtert - die Spitze des Dreiecks zeigt nach oben, zu einer *höheren* Zahl. Umgekehrt zeigt ein auf dem Kopf stehendes Dreieck an, dass das Risiko abnimmt (die Spitze zeigt also nach unten zu einer *niedrigeren* Zahl). Mit einem Kreis schließlich zeigen wir an, dass sich das Risiko weder verbessert noch verschlechtert. Das kann verwirrend sein, deshalb empfehlen wir, immer einen Schlüssel zu verwenden, wenn du ein Symbol zur Darstellung der Richtung verwendest.

RISK CRITERIA	Customer registration	Catalog checkout	Order fulfillment	Order shipment	TOTAL RISK
Scalability	2	6	1	2	11
Availability	3	4	2	1	10
Performance	4	2	3	6	15
Security	6	3	1	1	11
Data integrity	9	6	1	1	17
TOTAL RISK	24	21	8	11	

Abbildung 22-4. Darstellung der Risikorihtung durch Dreiecke

Diese überarbeitete architektonische Risikobewertung, die jetzt die Richtung angibt, erzählt eine andere Geschichte als die ursprüngliche. Erstens sehen wir, dass sich die Datenintegrität bei den kontinuierlichen

Messungen (das Dreieck zeigt nach oben) für den Katalog-Checkout, die Bestellabwicklung und den Bestellversand verschlechtert, was auf ein Datenbankproblem hindeuten könnte. Bei der Kundenregistrierung und dem Katalog-Checkout hingegen werden Sicherheit und Verfügbarkeit insgesamt besser (das Dreieck zeigt nach unten), was auf Verbesserungen in diesen Bereichen hinweist.

Als Nächstes beschreiben wir einen Prozess, den wir "*Risk Storming*" nennen und den Teams nutzen können, um *den* Risikograd für bestimmte Kontexte und Kriterien zu ermitteln.

## Risiko-Stürmung

Kein Architekt kann im Alleingang das Gesamtrisiko eines Systems bestimmen, und zwar aus zwei Gründen. Erstens könnte ein Architekt, der allein arbeitet, einen Risikobereich übersehen oder übersehen, und zweitens kennen nur sehr wenige Architekten *alle* Teile des Systems. An dieser Stelle kann Risk Storming helfen.

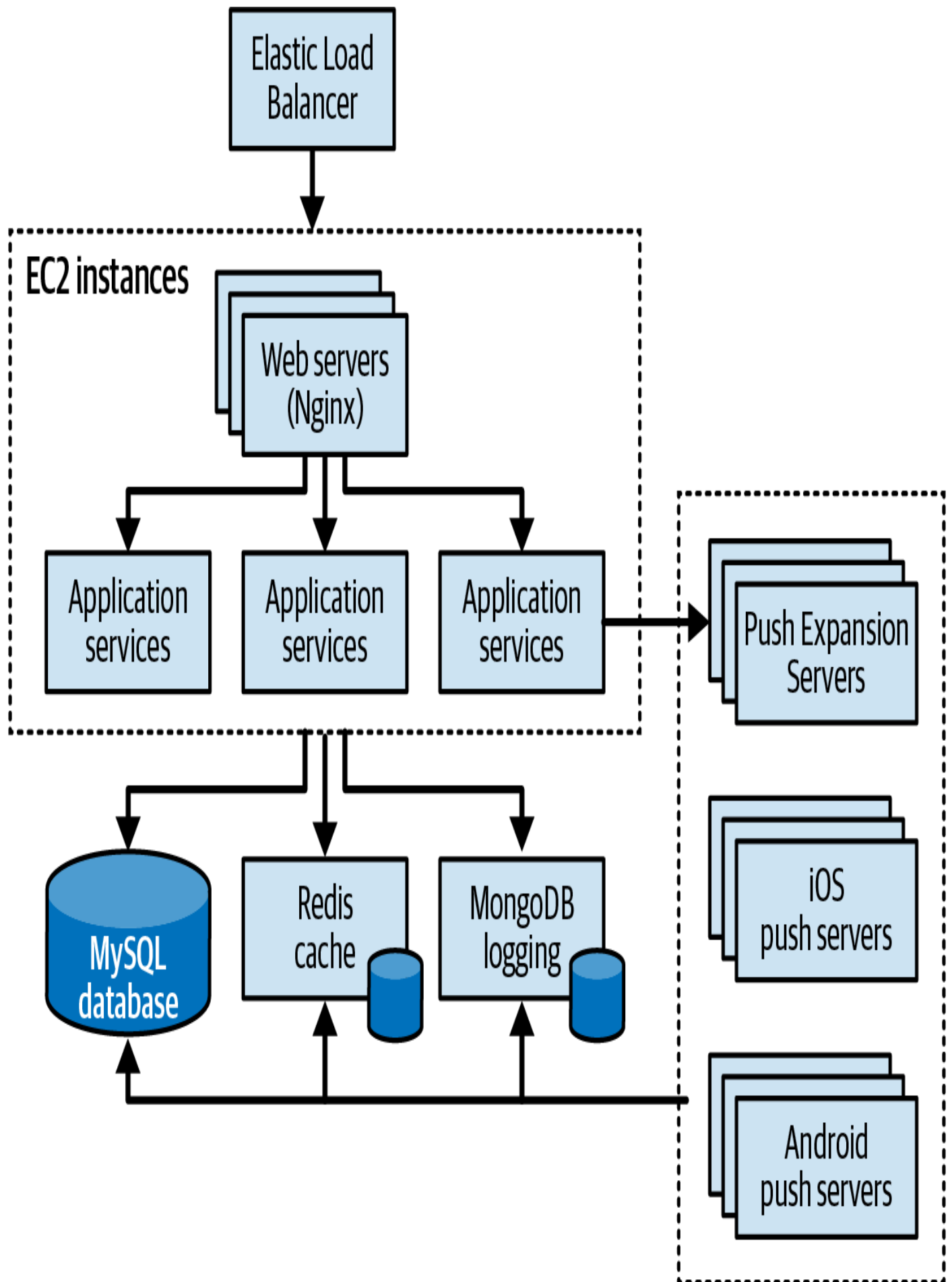
*Risk Storming* ist eine gemeinschaftliche Übung, um das architektonische Risiko innerhalb einer bestimmten Dimension (entweder Kontext oder Kriterien) zu bestimmen. An den meisten Risiko-Stormings sind mehrere Architekten beteiligt, aber wir empfehlen dringend, auch Senior-Entwickler und technische Leiter einzubeziehen. Sie bieten nicht nur eine Umsetzungsperspektive für das architektonische Risiko, sondern können durch ihre Beteiligung auch die Architektur besser verstehen.

Das Risk Storming umfasst drei Phasen: Identifizierung, Konsens und Minderung. In der individuellen Phase (Phase 1) verwenden alle Teilnehmer/innen die Risikomatrix, um den verschiedenen Bereichen der Architektur Risiken zuzuordnen. Diese individuelle Phase des Risk Storming ist wichtig, damit die Teilnehmer/innen die anderen Teilnehmer/innen nicht beeinflussen oder die Aufmerksamkeit der anderen von bestimmten Bereichen der Architektur ablenken. In den beiden kollaborativen Phasen arbeiten alle Teilnehmer/innen zusammen, um einen Konsens über die Risikobereiche zu erzielen und sie zu diskutieren (Phase 2) und Lösungen zur Risikominderung zu finden (Phase 3).

In allen drei Phasen wird ein umfassendes oder kontextbezogenes Architekturdiagramm verwendet (siehe [Kapitel 23](#)). Der Architekt, der das Risiko-Storming durchführt - wir nennen ihn den *Moderator* - ist dafür verantwortlich, dass alle Teilnehmer aktualisierte Diagramme für die Risiko-Storming-Sitzung erhalten.

[Abbildung 22-5](#) zeigt eine Beispiellarchitektur, die wir zur Veranschaulichung des Risikostorming-Prozesses verwenden werden. In dieser Architektur leitet ein Elastic Load Balancer eine Anfrage an jede EC2-Instanz weiter, die die Webserver (Nginx) und Anwendungsdienste enthält. Die Anwendungsdienste rufen eine MySQL-Datenbank, einen Redis-Cache und eine MongoDB-Datenbank (für die Protokollierung) auf. Sie rufen auch die Push Expansion Server auf, die wiederum mit der MySQL-Datenbank, dem Redis-Cache und der MongoDB-Protokollierung verbunden sind. (Mach dir keine Sorgen, wenn du all diese Produkte und

Schlagworte nicht verstehst - diese allzu vage, verallgemeinerte  
Architektur soll nur veranschaulichen, wie Risk Storming funktioniert).



Wir beginnen mit Phase 1.

## Phase 1: Identifizierung

In der *Identifizierungsphase* des Risk Storming identifiziert jeder Teilnehmer einzeln die Risikobereiche innerhalb der Architektur. In dieser Phase ist es wichtig, dass jeder Teilnehmer seine unvoreingenommene Sicht auf das Risiko festhält, ohne sich von anderen Teilnehmern beeinflussen zu lassen. Die Identifizierungsphase besteht aus drei Schritten:

1. Der Moderator schickt allen Teilnehmern eine Einladung zu den Kooperationsphasen. Die Einladung enthält das Architekturdiagramm (oder den Ort, an dem es zu finden ist), die zu analysierenden Risikokriterien und den Kontext sowie Datum, Uhrzeit und Ort (physisch oder virtuell) der kollaborativen Sitzung und alle weiteren logistischen Details.
2. Die Teilnehmer nutzen die Risikomatrix, um die Risiken der Architektur einzeln zu analysieren.
3. Die Teilnehmer/innen stufen jedes Risiko als gering (1-2), mittel (3-4) oder hoch (6-9) ein und schreiben die Zahlen auf kleine grüne, gelbe oder rote Klebezettel.

Bei den meisten Risiko-Stormings wird nur ein bestimmtes Kriterium oder ein bestimmter Kontext analysiert (z. B. "Wo liegen unsere Sicherheitsrisiken?" oder "Welche Bereiche sind bei der

Kundenregistrierung gefährdet?) Wenn es jedoch Probleme mit der Personalverfügbarkeit oder dem Zeitplan gibt, muss das Risikobewertungsteam möglicherweise mehrere Dimensionen innerhalb eines bestimmten Kontextes analysieren (z. B. Leistung und Skalierbarkeit). In diesen Fällen schreiben die Teilnehmer/innen das jeweilige Kriterium neben der Risikonummer auf die Klebezettel. Nehmen wir zum Beispiel an, dass drei Teilnehmer ein Risiko bei einer zentralen Datenbank sehen. Alle drei Teilnehmer stufen das Risiko als hoch (6) ein, aber ein Teilnehmer sieht es als Risiko für die Verfügbarkeit, während die anderen beiden es als Risiko für die Leistung sehen. Die Teilnehmer/innen sollten diese beiden Kriterien getrennt voneinander diskutieren.

---

#### TIPP

Wenn möglich, beschränke dich beim Risikostorming auf ein einziges Kriterium oder einen Kontext. So können sich die Teilnehmer/innen auf diese spezielle Dimension konzentrieren und es wird vermieden, dass sie das tatsächliche Risiko nicht richtig einschätzen können.

---

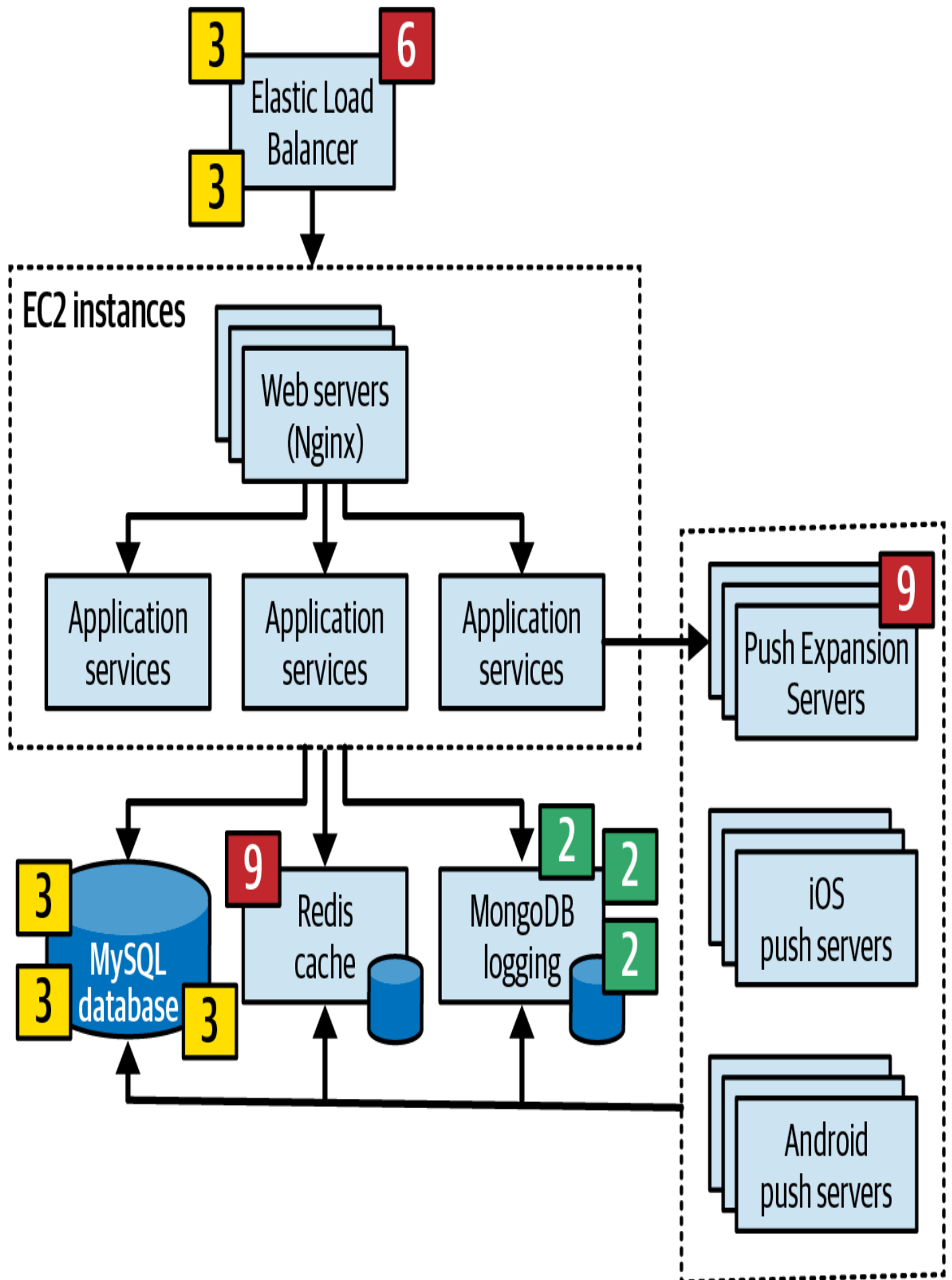
## Phase 2: Konsens

Die *Konsensphase* des Risk Storming ist sehr kooperativ und zielt darauf ab, einen Konsens zwischen allen Teilnehmern über das Risiko bzw. die Risiken innerhalb der Architektur zu erzielen. Diese Aktivität ist am effektivsten, wenn der Moderator ein großes gedrucktes Architekturdiagramm an die Wand hängt (oder eine elektronische Version auf einem großen Bildschirm). Wenn die Teilnehmer/innen beim

Risiko-Storming ankommen, weist der/die Moderator/in sie an, ihre Klebezettel mit den Risikostufen auf dem entsprechenden Bereich des Architekturdiagramms zu platzieren (siehe [Abbildung 22-6](#)).

Sobald alle Haftnotizen vorhanden sind, kann die Phase der Zusammenarbeit beginnen. Hier geht es darum, die Risikobereiche im Team zu analysieren und einen Konsens über den Risikograd zu finden. In dem in [Abbildung 22-6](#) dargestellten Beispiel hat das Team mehrere Risikobereiche identifiziert. (Die tatsächlichen Kriterien sind für dieses Beispiel nicht wichtig.) Das können wir sehen:

- Zwei Teilnehmer stuften den Elastic Load Balancer als mittleres Risiko ein (3), während ein Teilnehmer ihn als hohes Risiko einstuft (6).
- Ein Teilnehmer bezeichnete die Push Expansion Server als hohes Risiko (9).
- Drei Teilnehmer stuften die MySQL-Datenbank als mittleres Risiko ein (3).
- Ein Teilnehmer stuft den Redis-Cache als hohes Risiko ein (9).
- Drei Teilnehmer stuften die MongoDB-Protokollierung als geringes Risiko ein (2).
- Niemand hat andere Bereiche der Architektur als risikobehaftet identifiziert, also gibt es auch keine Haftnotizen für andere Bereiche.



Die MySQL-Datenbank und die MongoDB-Protokollierung müssen in dieser Sitzung nicht weiter diskutiert werden, da sich alle Teilnehmer über ihre Risikostufen einig sind. Beim Elastic Load Balancer gehen die Meinungen jedoch auseinander, und die Push Expansion Server und der Redis-Cache wurden nur von jeweils einem Teilnehmer als Risiko eingestuft. Um diese Unstimmigkeiten zu beseitigen, ist die kollaborative Phase genau das Richtige.

Zwei Teilnehmer (Austen und Logan) stufen den Elastic Load Balancer als mittleres Risiko ein (3), einer (Addison) als hohes Risiko (6). Austen und Logan fragen Addison, warum sie das Risiko als hoch einstufen. Addison antwortet, dass bei einem Ausfall des Elastic Load Balancers das gesamte System unzugänglich wird. Das ist zwar richtig und erhöht das *Risiko* auf hoch, aber die beiden anderen Teilnehmer überzeugen Addison davon, dass das Risiko durch das Clustering gering ist. Addison stimmt dem zu und die Gruppe senkt die *Wahrscheinlichkeitsstufe* auf mittel (3).

Das hätte aber auch anders ausgehen können. Wenn Austen und Logan einen bestimmten Aspekt des Risikos im Elastic Load Balancer übersehen hätten, den Addison gesehen hat, hätte Addison die anderen beiden Teilnehmer davon überzeugen können, dieses Risiko als hoch statt als mittel einzustufen. Deshalb ist die Phase der Zusammenarbeit beim Risk Storming so wichtig.

Ein Teilnehmer stufte die Push Expansion Server als hohes Risiko ein (9), aber kein anderer Teilnehmer sah in diesem Bereich der Architektur überhaupt ein Risiko. Die Person, die das Risiko identifiziert hat, erklärt, dass sie das Risiko als hoch einstuft, weil sie schlechte Erfahrungen mit Push-Expansionsservern gemacht hat, die bei hohen Belastungen, wie sie bei dieser Architektur auftreten, ständig abstürzen. Dieses Beispiel zeigt den Wert des Risk Storming - ohne die Beteiligung dieses Teilnehmers hätte niemand das hohe Risiko erkannt, bis es in der Produktion angekommen wäre.

Der Redis-Cache ist ein interessanter Fall. Ein Teilnehmer, ein Entwickler namens Devon, stufte ihn als hohes Risiko ein (9), aber niemand sonst sah in diesem Cache ein Risiko. Als die anderen Teilnehmer Devon fragen, warum sie dieses Risiko als hoch einstufen, antwortet er: "Was ist ein Redis-Cache?" Immer wenn ein Teilnehmer des Risiko-Stormings eine Technologie als unbekannt bezeichnet, wird dieser Bereich automatisch als hohes Risiko eingestuft (9).

---

#### TIPP

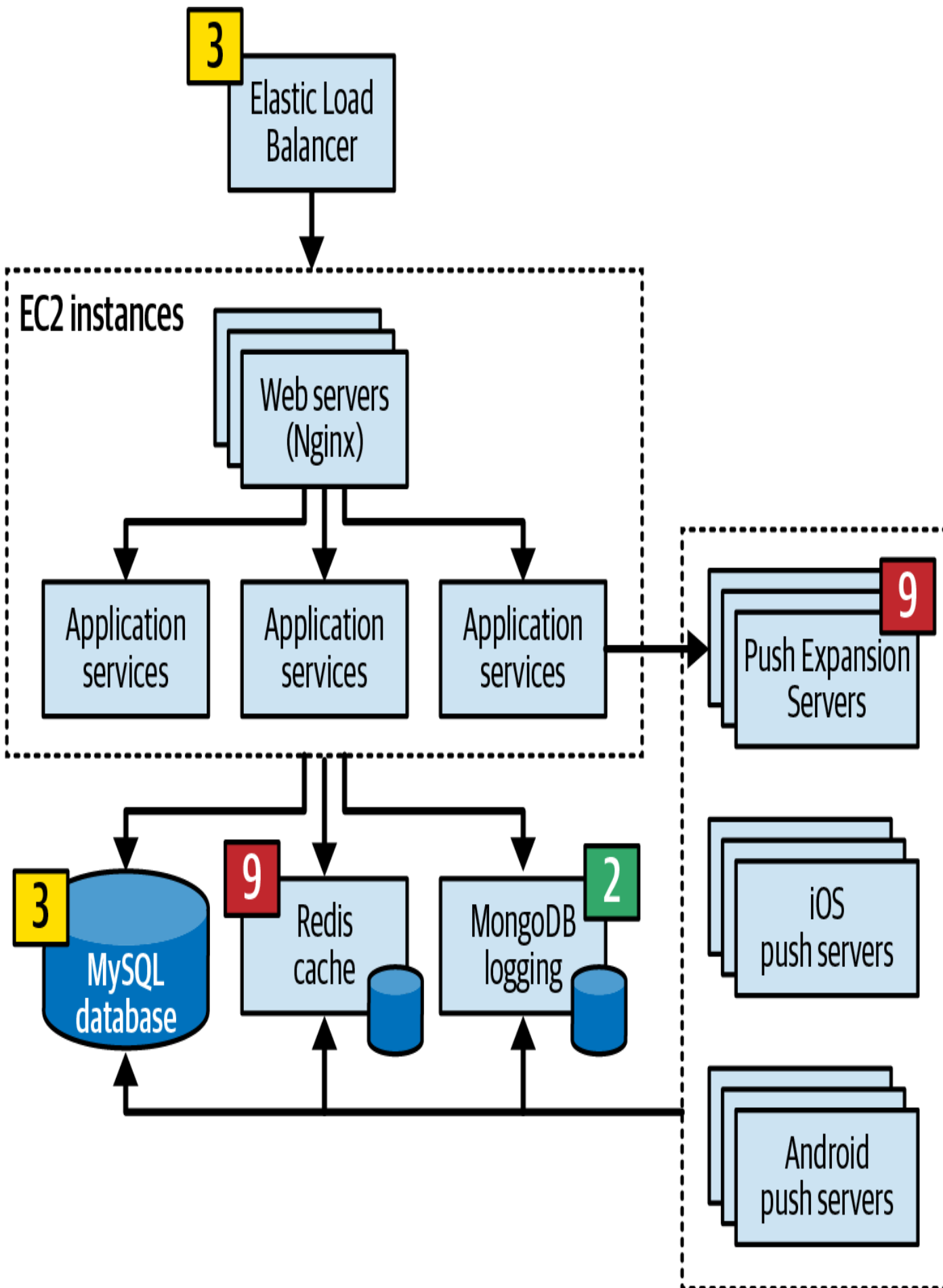
Weise unbewährten oder unbekannten Technologien immer die höchste Risikobewertung (9) zu, da die Risikomatrix für dieses Kriterium oder diesen Kontext nicht verwendet werden kann.

---

Das Beispiel des Redis-Caches verdeutlicht, warum es wichtig ist, die Entwickler in die Risikobesprechungen einzubeziehen. Die Tatsache, dass dieser Teilnehmer eine bestimmte Technologie nicht kannte, ist für den Architekten eine wertvolle Information über das Gesamtrisiko. Der

Architekt könnte beschließen, die Technologie zu ändern oder Schulungskosten zu übernehmen, um das Entwicklungsteam auf den neuesten Stand zu bringen.

Diese Phase wird fortgesetzt, bis sich alle Teilnehmer/innen über die ermittelten Risikobereiche einig sind. Sobald alle Haftnotizen konsolidiert sind, endet diese Phase. Das Endergebnis ist in Abbildung 22-7 dargestellt.



## Phase 3: Risikominderung

Sobald sich alle Beteiligten über die Risikostufen der Architektur einig sind, beginnt die Phase der *Risikominderung*. Zur Risikominderung müssen in der Regel bestimmte Bereiche der Architektur geändert werden, die ansonsten als perfekt angesehen worden wären.

In dieser Phase, die ebenfalls kollaborativ abläuft, wird nach Möglichkeiten gesucht, die in der zweiten Phase ermittelten Risiken zu verringern oder zu beseitigen. Abhängig von den ermittelten Risiken muss die ursprüngliche Architektur möglicherweise komplett geändert werden, oder die Änderungen beschränken sich auf ein einfaches Refactoring der Architektur in bestimmten Bereichen, wie z. B. das Hinzufügen einer Warteschlange für den Gegendruck, um einen Durchsatzengpass zu verringern.

Unabhängig davon, welche Änderungen erforderlich sind, verursacht die Phase der Risikominderung in der Regel zusätzliche Kosten. Deshalb ist es wichtig, dass in dieser Phase die wichtigsten Interessengruppen des Unternehmens einbezogen werden, die entscheiden können, ob die Kosten einer bestimmten Lösung zur Risikominderung das Risiko überwiegen.

Nehmen wir zum Beispiel an, dass das Team in unserem Beispiel-Risiko-Storming die zentrale Datenbank als mittleres Risiko (4) in Bezug auf die allgemeine Systemverfügbarkeit einstuft. Die Teilnehmer sind sich einig,

dass ein Clustering der Datenbank und ihre Aufteilung in separate physische Datenbanken dieses Risiko mindern würde. Diese Lösung würde jedoch 50.000 \$ kosten. Der moderierende Architekt trifft sich mit den wichtigsten Interessenvertretern des Unternehmens, einschließlich des Eigentümers, um die Kompromisse zwischen Verfügbarkeitsrisiko und Kosten zu diskutieren. Der Geschäftsinhaber entscheidet, dass der Preis zu hoch ist und dass die Kosten das Verfügbarkeitsrisiko nicht aufwiegen. Der Architekt schlägt daraufhin einen anderen Ansatz vor: Wie wäre es, die Datenbank in zwei separate domänenbasierte Datenbanken aufzuteilen, anstatt ein teures Clustering durchzuführen? Diese Lösung würde nur 16.000 \$ kosten und trotzdem das Verfügbarkeitsrisiko verringern. Die Beteiligten stimmen diesem Kompromiss zu.

Dieses Szenario zeigt, wie das Risk Storming nicht nur die Gesamtarchitektur, sondern auch die Verhandlungen zwischen Architekten und Stakeholdern beeinflusst. In Kombination mit den Risikobewertungen, die wir zu Beginn dieses Kapitels beschrieben haben, ist das Risk Storming ein hervorragendes Mittel, um Risiken zu identifizieren und zu verfolgen, die Architektur zu verbessern und die Verhandlungen zwischen den wichtigsten Interessengruppen zu strukturieren.

## User-Story Risikoanalyse

Risk Storming ist nicht nur bei der Identifizierung von architektonischen Risiken nützlich, sondern auch bei vielen anderen Aspekten der

Softwareentwicklung. Ein Entwicklungsteam kann das Risk Storming zum Beispiel nutzen, um das Gesamtrisiko für die Fertigstellung einer Benutzergeschichte in einer bestimmten Iteration (und damit die Gesamtrisikobewertung dieser Iteration) während des Story Groomings zu ermitteln. Mithilfe der gleichen Risikomatrix kann das Team das Risiko einer Benutzergeschichte ermitteln, indem es die Gesamtauswirkungen ermittelt, wenn die Geschichte nicht innerhalb der Iteration fertiggestellt wird, und die Wahrscheinlichkeit, dass die Geschichte in der aktuellen Iteration nicht fertiggestellt wird. Dann kann das Team risikoreiche Stories identifizieren, sie sorgfältig verfolgen und besser priorisieren.

## Risk-Storming Anwendungsfall

Um zu verdeutlichen, wie leistungsfähig Risk Storming ist und wie es die Gesamtarchitektur verbessern kann, betrachten wir das Beispiel eines Supportsystems für ein Callcenter, in dem Krankenschwestern und -pfleger Patienten zu verschiedenen Gesundheitszuständen beraten. Die Systemanforderungen sind:

- Eine Drittanbieter-Diagnose-Engine wird Fragen stellen und Krankenschwestern und Patienten durch ihre medizinischen Probleme führen. Dieses System kann etwa 500 Anfragen pro Sekunde bearbeiten.
- Die Patienten können entweder im Call Center anrufen, um mit einer Krankenschwester zu sprechen, oder eine Selbstbedienungs-Website nutzen, die direkt auf dieselbe Diagnosemaschine zugreift.

- Das System muss landesweit 250 Krankenschwestern und Krankenpfleger und bis zu Hunderttausende von Selbstbedienungspatienten gleichzeitig unterstützen.
- Krankenschwestern und -pfleger können über einen Austausch von Krankenakten auf die Krankenakten der Patienten zugreifen, aber die Patienten können nicht auf ihre eigenen Krankenakten zugreifen.
- Das System muss dem HIPAA (Health Insurance Portability and Accountability Act) entsprechen, d.h. es ist wichtig, dass niemand außer den Krankenschwestern und -pflegern auf die Krankenakten der Patienten zugreifen kann. Die Selbstbedienungsoption kann die Einhaltung des HIPAA nicht garantieren.
- Das System muss in der Lage sein, bei Erkältungs-, Grippe- und COVID-Ausbrüchen ein hohes Volumen zu bewältigen.
- Die Anrufe werden je nach Qualifikationsprofil der Krankenschwestern und Krankenpfleger (z. B. gesprochene Sprachen oder medizinische Spezialisierungen) an diese weitergeleitet.

Nach der Analyse dieser Anforderungen entwirft Logan, der für dieses System verantwortliche Architekt, die in Abbildung 22-8 dargestellte High-Level-Architektur. Diese Architektur besteht aus drei separaten webbasierten Benutzeroberflächen: eine für die Selbstbedienung, eine für die Krankenschwestern, die Anrufe entgegennehmen, und eine für das Verwaltungspersonal, um Krankenschwesterprofile und Konfigurationseinstellungen hinzuzufügen und zu pflegen. Der Callcenter-Teil des Systems besteht aus dem Dienst Call Acceptor, der Anrufe entgegennimmt, und dem Dienst Call Router, der den Anrufer auf der Grundlage des Qualifikationsprofils der Krankenschwester an die

nächste verfügbare Schwester weiterleitet. Der Call Router Dienst greift auf die zentrale Datenbank zu, um Informationen über das Profil der Krankenschwester zu erhalten. Das Herzstück dieser Architektur ist ein API-Gateway für das Diagnosesystem, das Sicherheitsprüfungen durchführt und Anfragen an den entsprechenden Backend-Dienst weiterleitet.

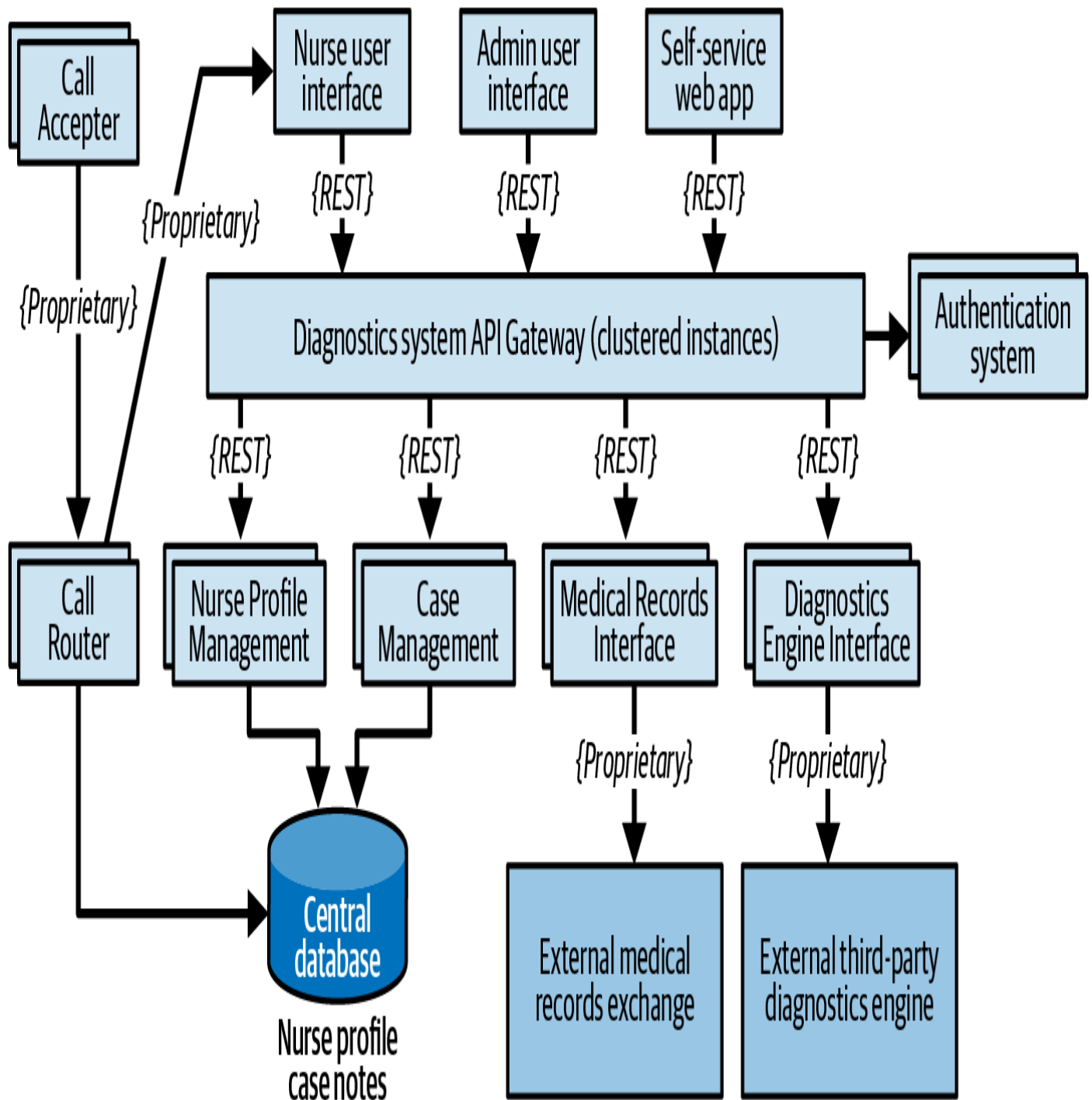


Abbildung 22-8. High-Level-Architektur für ein Pflege-Hotline-Diagnosesystem

Die vier wichtigsten Dienste in diesem System sind der Case Management Dienst, der Nurse Profile Management Dienst, der Medical Records Interface Dienst zum Austausch medizinischer Daten und der Diagnostics Engine Interface Dienst eines externen

Anbieters. Die gesamte Kommunikation erfolgt über REST, mit Ausnahme der proprietären Protokolle zu den externen Systemen und den Callcenter-Diensten. Die Bereiche, die die Architektur unterstützen muss, sind, zusammenfassend gesagt, Verfügbarkeit, Elastizität und Sicherheit.

Nach vielen Überprüfungen ist Logan der Meinung, dass die Architektur bereit für die Umsetzung ist. Als verantwortungsbewusster und effektiver Architekt beschließt Logan jedoch, ein Risikostorming durchzuführen.

## Verfügbarkeit

Als Moderator beschließt Logan, das erste Risiko-Storming auf die Verfügbarkeit zu konzentrieren, die für den Erfolg des Systems entscheidend ist. Nach der Identifizierungs- und Kooperationsphase kommen die Teilnehmer/innen zu folgenden Risikobereichen (siehe [Abbildung 22-9](#)):

- Verfügbarkeit der zentralen Datenbank: hohes Risiko (6) aufgrund der hohen Auswirkungen (3) und der mittleren Wahrscheinlichkeit (2), dass die Datenbank bei Bedarf nicht verfügbar ist.
- **Diagnostics Engine** Verfügbarkeit: hohes Risiko (9) aufgrund der hohen Auswirkungen (3) und der unbekannten Wahrscheinlichkeit (3) der Nichtverfügbarkeit.
- **Medical Records Interface** Verfügbarkeit: geringes Risiko (2); diese Komponente ist nicht erforderlich, um ein bestimmtes medizinisches Ergebnis zu bestimmen.

- Das Team hat keine anderen Teile des Systems als Verfügbarkeitsrisiko eingestuft, da die Architektur mehrere Instanzen jedes Dienstes und Cluster des API-Gateways umfasst.

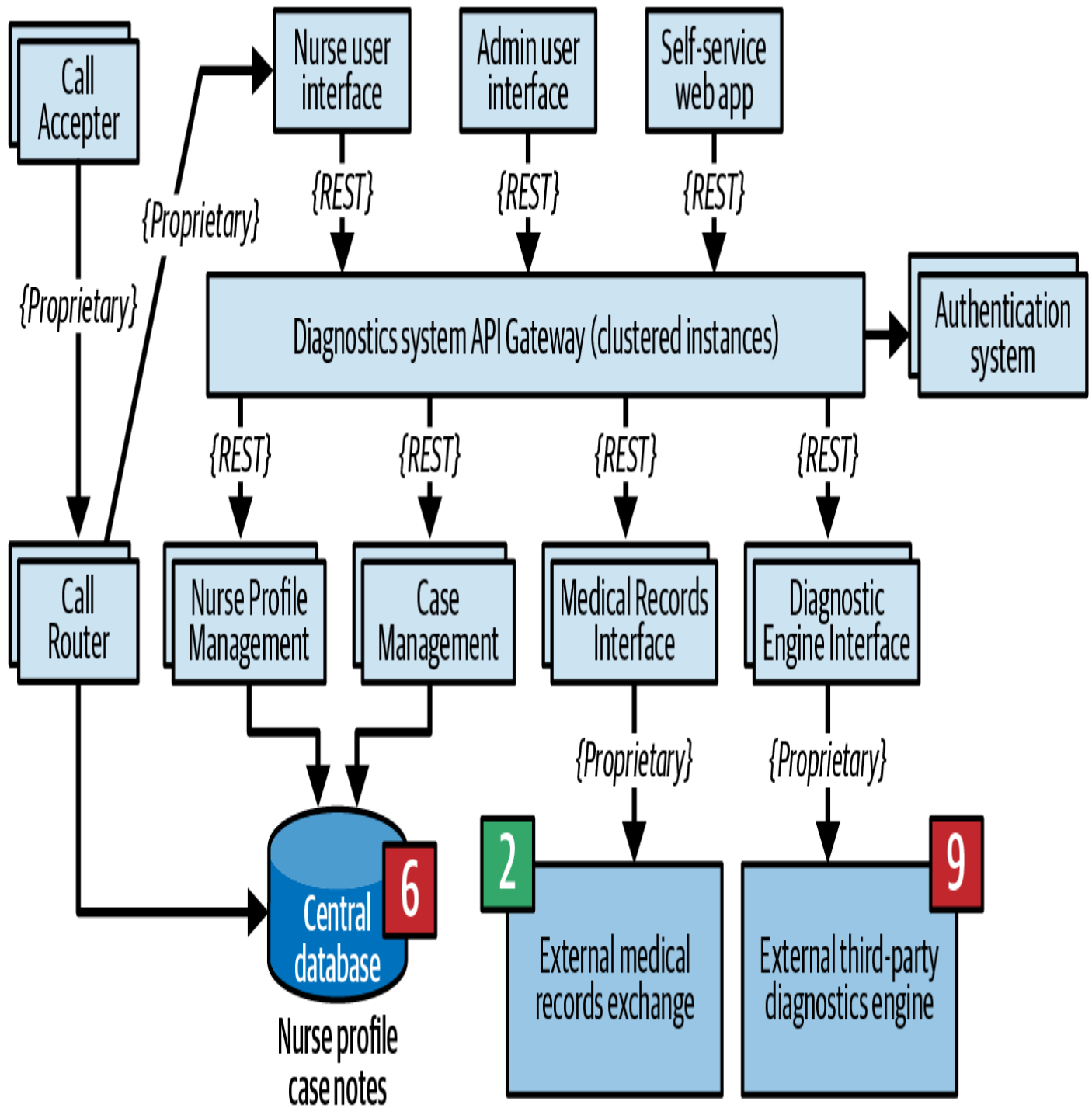


Abbildung 22-9. Risikobereiche für die Verfügbarkeit, wie vom Risk-Storming-Team ermittelt

Alle Beteiligten sind sich einig, dass bei einem Ausfall der Datenbank die Krankenschwestern die Fallnotizen manuell aufschreiben könnten, der Call Router aber nicht funktionieren würde. Um dieses Risiko zu minimieren, beschließen sie gemeinsam, die physische Datenbank in zwei separate Datenbanken aufzuteilen: eine Cluster-Datenbank, die die Schwesternprofilinformationen enthält, und eine Single-Instance-Datenbank für die Fallnotizen. Mit dieser Änderung der Architektur werden nicht nur die Probleme der Datenbankverfügbarkeit gelöst, sondern auch die Fallnotizen gesichert.

Es ist viel schwieriger, das Verfügbarkeitsrisiko bei externen Systemen (in diesem Fall **Diagnostics Engine** und **Medical Records Interface**) zu mindern, da sie von einer dritten Partei kontrolliert werden. Das Team beschließt zu recherchieren, ob diese Systeme Service Level Agreements (SLAs) oder Service Level Objectives (SLOs) veröffentlicht haben. Ein SLA ist in der Regel eine rechtsverbindliche vertragliche Vereinbarung; ein SLO ist normalerweise nicht rechtsverbindlich. Sie finden SLAs für beide Systeme. Das SLA von **Diagnostics Engine** garantiert eine Verfügbarkeit von 99,99% (das sind 52,60 Minuten Ausfallzeit pro Jahr), und das von **Medical Records Interface** garantiert eine Verfügbarkeit von 99,90% (das sind 8,77 Stunden Ausfallzeit pro Jahr). Diese Informationen reichten dem Risikobewertungsteam aus, um das festgestellte Risiko zu beseitigen.

Nach diesem Risiko-Storming ändert das Team die Architektur, wie in [Abbildung 22-10](#) dargestellt, erstellt zwei Datenbanken und fügt die SLAs zum Architekturdiagramm hinzu.

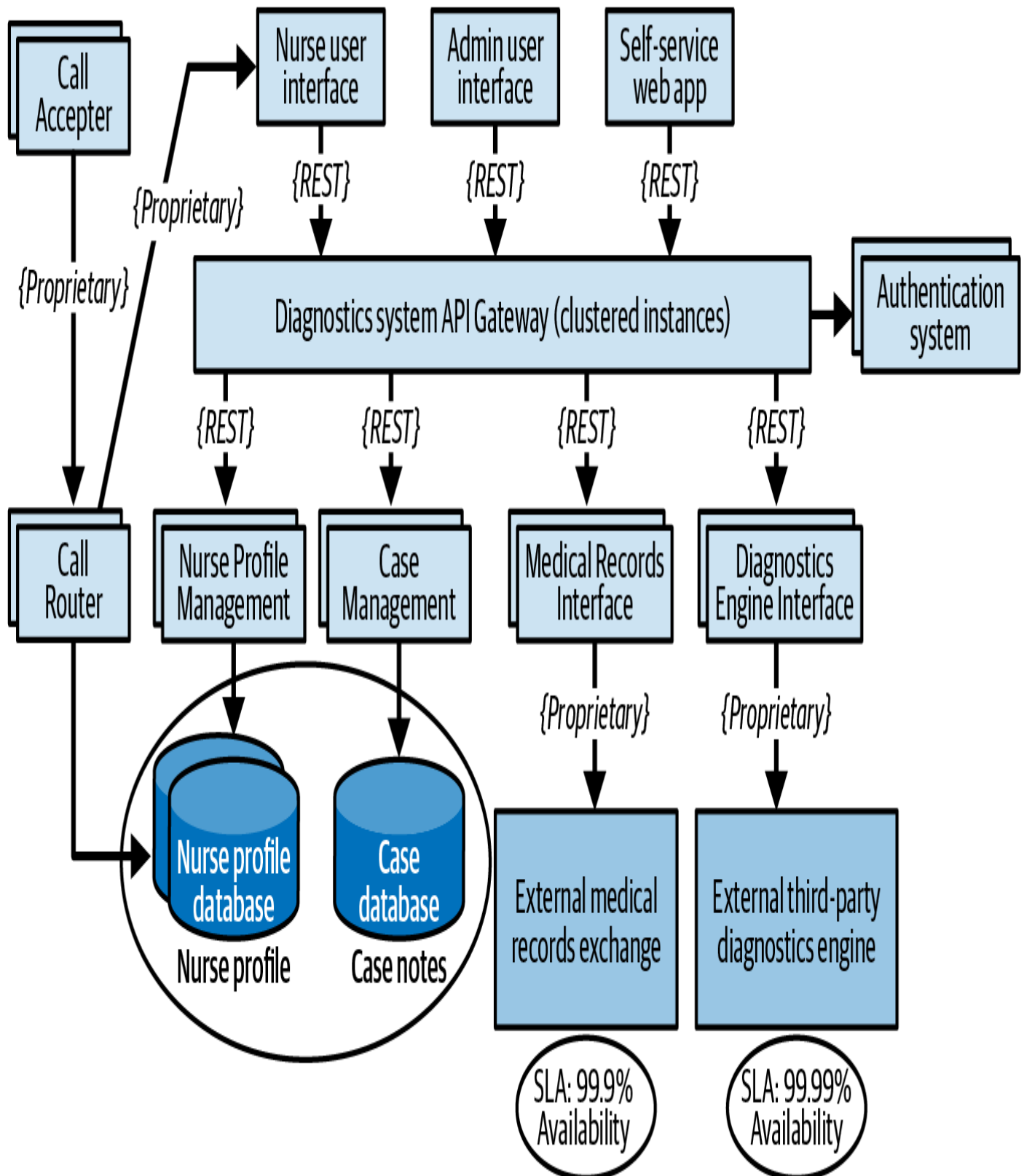


Abbildung 22-10. Verfügbarkeitsrisikobereiche können durch die Verwendung separater Datenbanken gemildert werden

## Elastizität

Die zweite Risikoübung konzentriert sich auf die Elastizität - Spitzen in der Benutzerlast (auch bekannt als variable Skalierbarkeit). Obwohl es nur 250 Krankenschwestern gibt (was bedeutet, dass nicht mehr als 250 Krankenschwestern auf **Diagnostics Engine** zugreifen werden), kann der Selbstbedienungsbereich des Systems auch auf **Diagnostics Engine** zugreifen, wodurch die Anzahl der Anfragen an die **Diagnostics Engine** Schnittstelle erheblich steigt. Die Teilnehmer des Risiko-Stormings sind besorgt über die Grippesaison und COVID-Ausbrüche, die die zu erwartende Belastung des Systems deutlich erhöhen werden.

Die Teilnehmer stufen die Schnittstelle **Diagnostics Engine** einstimmig als hohes Risiko ein (9). Da sie nur 500 Anfragen pro Sekunde bewältigen kann, rechnen sie zu Recht mit einem hohen Risiko, dass sie mit dem erwarteten Durchsatz nicht mithalten kann, insbesondere mit REST als Schnittstellenprotokoll.

Das Team hat beschlossen, dass eine Möglichkeit, dieses Risiko zu mindern, darin besteht, asynchrone Warteschlangen (Messaging) für die Kommunikation zwischen dem API Gateway und der Schnittstelle **Diagnostics Engine** zu verwenden. Dadurch entsteht ein Rückstau, falls die Aufrufe an die **Diagnostics Engine** zurückbekommen. Das ist zwar eine gute Praxis, aber das Risiko wird dadurch nicht ganz gemindert: Krankenschwestern und Selbstbedienungspatienten werden immer noch zu lange auf Antworten von **Diagnostics Engine** warten müssen, und ihre Anfragen werden wahrscheinlich nicht rechtzeitig beantwortet.

Die Teilnehmer beschließen, ein Muster zu verwenden, das als Ambulance-Muster bekannt ist, um diese Anfragen zu trennen, indem sie zwei Nachrichtenkanäle statt nur einem verwenden. So kann das System die Anfragen der Krankenschwestern gegenüber den Selbstbedienungsanfragen priorisieren. Das würde das Risiko mindern, aber die Wartezeiten würden dadurch nicht verringert. Nach weiteren Diskussionen beschließt die Gruppe, die Zahl der ausbruchsbedingten Anrufe auf **Diagnostics Engine** zu reduzieren, indem diese speziellen Diagnosefragen zwischengespeichert werden, damit sie nie die Schnittstelle **Diagnostics Engine** erreichen.

Neben der Einrichtung von zwei Nachrichtenkanälen (einer für die Krankenschwestern und einer für die Selbstbedienungspatienten) erstellt das Team einen neuen Dienst namens **Diagnostics Outbreak Cache Server**, der alle Anfragen zu einem bestimmten Ausbruch oder einer grippebezogenen Frage bearbeitet (Abbildung 22-11). Diese neue Architektur reduziert die Anzahl der Aufrufe an die Diagnose-Engine und ermöglicht mehr gleichzeitige Anfragen zu anderen Symptomen. Ohne das Risiko-Storming wäre dieses Risiko vielleicht erst in der Grippezeit erkannt worden.

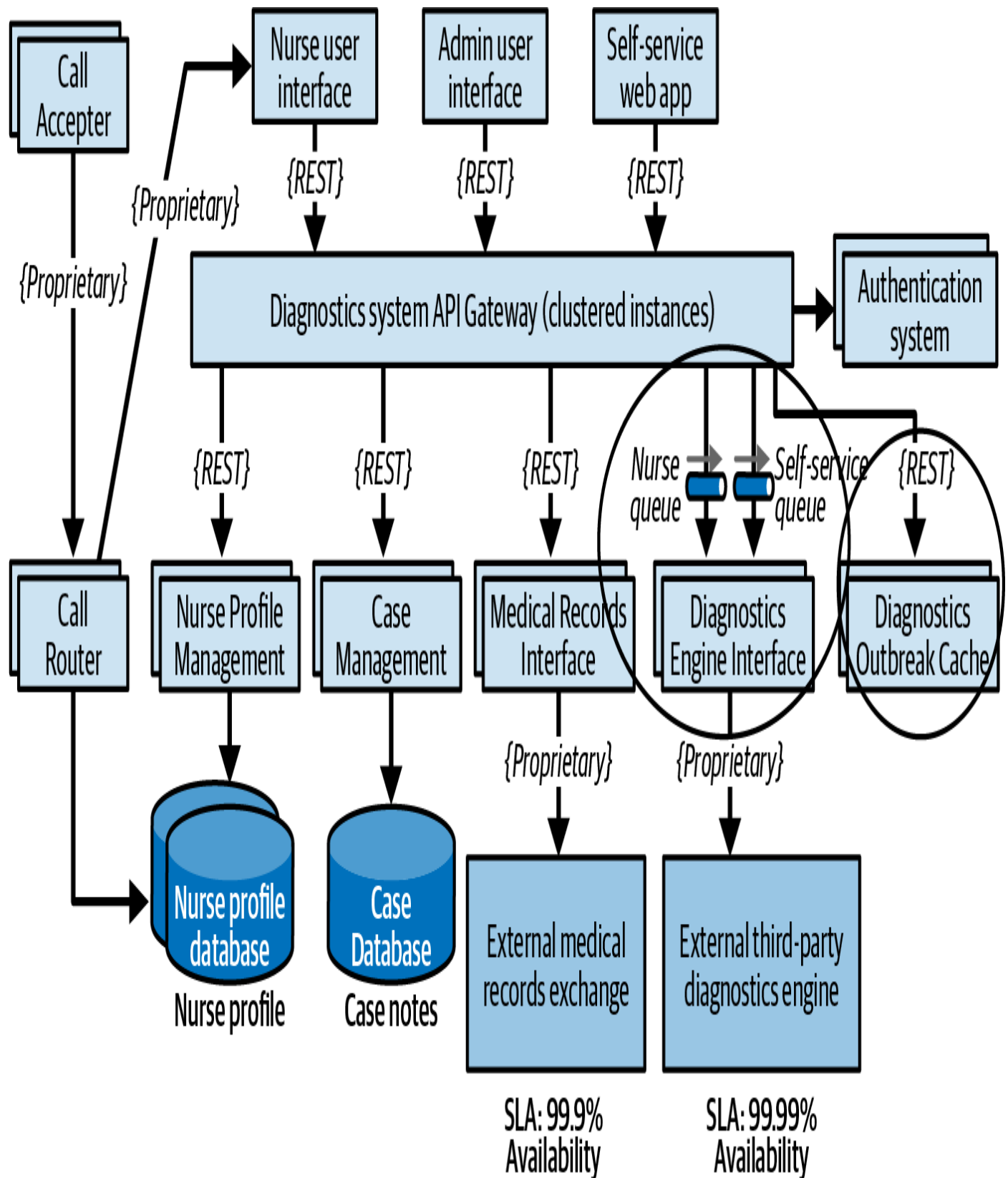


Abbildung 22-11. Änderungen an der Architektur, um dem Elastizitätsrisiko zu begegnen

## Sicherheit

Ermutigt durch diese Erfolge beschließt der Architekt, eine letzte Risikositzung zu veranstalten, die sich auf ein weiteres entscheidendes Merkmal für dieses System konzentriert - die Sicherheit. Aufgrund der HIPAA-Bestimmungen darf **Medical Records Interface** nur Krankenschwestern und -pflegern den Zugriff auf die Krankenakten der Patienten erlauben. Der Architekt glaubt, dass die Authentifizierungs- und Autorisierungsprüfungen im API-Gateway dieses Risiko neutralisieren, aber er ist neugierig, ob die Teilnehmer noch andere Sicherheitsrisiken finden werden.

Alle Teilnehmer bezeichnen das API-Gateway des Diagnosesystems als hohes Sicherheitsrisiko (6). Sie geben an, dass es große Auswirkungen hat, wenn Verwaltungsmitarbeiter oder Selbstbedienungspatienten auf Krankenakten zugreifen (3), schätzen die Wahrscheinlichkeit aber als mittel ein (2). Die Sicherheitsprüfungen für jeden einzelnen API-Aufruf sind zwar hilfreich, aber alle Aufrufe (Selbstbedienung, Verwaltung und Krankenschwestern) laufen immer noch über dasselbe API-Gateway. Sie überzeugen den Vermittler, der dieses Risiko zunächst als gering (2) eingestuft hatte, schließlich davon, dass das Risiko in Wirklichkeit hoch ist und gemindert werden muss.

Alle sind sich einig, dass getrennte API-Gateways für jeden Nutzertyp (Verwaltungspersonal, Selbstbedienungsnutzer und Krankenschwestern) verhindern würden, dass Anrufe, die nicht von Krankenschwestern kommen, jemals die **Medical Records Interface** erreichen. Die endgültige Version der Architektur des Architekten ist in [Abbildung 22-12](#) dargestellt.

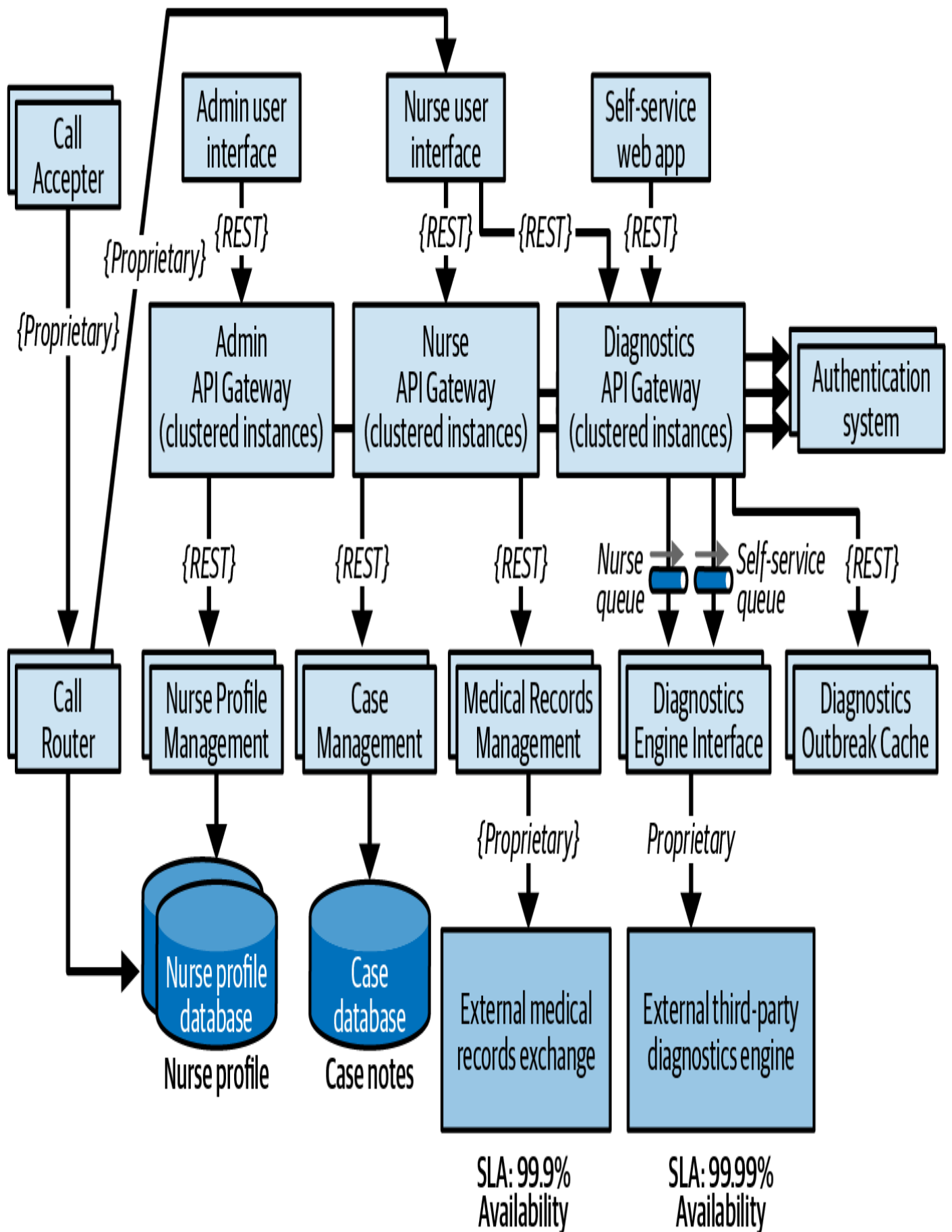


Abbildung 22-12. Änderungen an der Architektur zur Bewältigung des Sicherheitsrisikos

Dieses Beispiel verdeutlicht die Macht des Risk Storming. Architekten, Entwickler und wichtige Stakeholder arbeiten zusammen, betrachten die für den Erfolg des Systems wichtigsten architektonischen Merkmale und identifizieren Risikobereiche, die sonst unbemerkt geblieben wären.

Die ursprüngliche Architektur([Abbildung 22-8](#)) wird nach dem Risk Storming([Abbildung 22-12](#)) in einer Weise verändert, die Bedenken hinsichtlich Verfügbarkeit, Elastizität und Sicherheit ausräumt und diese Architektur effektiver und erfolgversprechender macht.

## Zusammenfassung

Risk Storming ist kein einmaliger Prozess. Er setzt sich während des gesamten Lebenszyklus eines Systems fort, da die Teams daran arbeiten, Risikobereiche zu identifizieren und zu entschärfen, bevor sie in der Produktion auftreten. Wie oft eine Organisation Risk Storming-Sitzungen abhalten sollte, hängt von vielen Faktoren ab, z. B. von der Häufigkeit der Änderungen, den Bemühungen um eine Überarbeitung der Architektur und der schrittweisen Entwicklung der Architektur. Es ist üblich, nach dem Hinzufügen einer wichtigen Funktion oder am Ende jeder Iteration ein Risiko-Storming zu bestimmten Dimensionen durchzuführen, um sicherzustellen, dass die Architektur korrekt ist und den Anforderungen des Unternehmens entspricht.